

# Distributed Evolution Strategies for Black-box Stochastic Optimization

Xiaoyu He, Zibin Zheng, Chuan Chen, Yuren Zhou, Chuan Luo, and Qingwei Lin

**Abstract**—This work concerns the evolutionary approaches to distributed stochastic black-box optimization, in which each worker can individually solve an approximation of the problem with nature-inspired algorithms. We propose a distributed evolution strategy (DES) algorithm grounded on a proper modification to evolution strategies, a family of classic evolutionary algorithms, as well as a careful combination with existing distributed frameworks. On smooth and nonconvex landscapes, DES has a convergence rate competitive to existing zeroth-order methods, and can exploit the sparsity, if applicable, to match the rate of first-order methods. The DES method uses a Gaussian probability model to guide the search and avoids the numerical issue resulted from finite-difference techniques in existing zeroth-order methods. The DES method is also fully adaptive to the problem landscape, as its convergence is guaranteed with any parameter setting. We further propose two alternative sampling schemes which significantly improve the sampling efficiency while leading to similar performance. Simulation studies on several machine learning problems suggest that the proposed methods show much promise in reducing the convergence time and improving the robustness to parameter settings.

**Index Terms**—Evolution strategies, distributed optimization, black-box optimization, stochastic optimization, zeroth-order methods.

## 1 INTRODUCTION

We consider the following stochastic optimization problem:

$$\min_{\boldsymbol{x} \in \mathbb{R}^n} f(\boldsymbol{x}) = \mathbb{E}[F(\boldsymbol{x}; \xi)] \quad (1)$$

where  $\boldsymbol{x} \in \mathbb{R}^n$  is the decision vector,  $\xi$  is a random variable,  $F$  is an unconstrained real-valued function, and  $\mathbb{E}[\cdot]$  denotes the expectation taken over the distribution of  $\xi$ . Problems of this type have a long history dating back to 1950's [1] and are still at the heart of many modern applications in machine learning [2], [3], signal processing [4], and automatic control [5]. For example, we can let  $\xi$  be a data point and  $F$  a loss assessing how  $\xi$  fits to a statistic model parametrized by  $\boldsymbol{x}$ ; the problem (1), in this way, then provides a universal formulation that captures a wide range of machine learning tasks [6]. The hardness of stochastic optimization mainly comes from the inherent noise nature, the possibly high dimensionality, and the complexity of objective landscapes. Despite this hardness, significant progress in the resolution of problem (1) has been made via exploring the gradient (first-order) or Hessian (second-order) information of the component function  $F$ . A variety of first-order and second-order stochastic optimization methods have been developed in recent years, enjoying both the theoretical and practical benefits; see [7], [8] for a comprehensive survey. However, when the landscape characteristics (differentiability,

smoothness, convexity, etc.) are unknown, stochastic optimization remains a challenging task.

In this paper, we are particularly interested in solving problem (1) in distributed black-box settings. Concretely, there are  $M$  workers having access to the distribution of  $\xi$ , but, for a given  $\boldsymbol{x}$ , they can only evaluate the stochastic objective value  $F(\boldsymbol{x}; \xi)$ . The workers may run individually and exchange information periodically through a parameter server, so they can minimize  $f$  in a collaborative manner. But apart from the decision vector  $\boldsymbol{x}$ , what they can share during the collaboration is limited to the function values (zeroth-order information), excluding the sharing of gradient or curvature information (which is case of existing first-/second-order distributed methods). The consideration of this setting is motivated by two scenarios in the real world. The first scenario is related to the on-device machine learning, sometimes referred to as federated learning [9], [10] or edge intelligence [11]. It is known that machine learning practitioners seldom derive gradients manually; instead, they rely on automatic differentiation [12] which computes the gradient during the function evaluation using the chain rule. The automatic differentiation tools work well on usual PCs, but they have a relatively large memory cost which may be prohibitive on mobile devices. In addition, automatic differentiation only runs on certain software environment and may cause compatibility issues in distributed settings.

The second applicable scenario is the parallel solving of stochastic black-box problems. Consider minimizing a time-consuming black-box function defined over a massive amount of data and the goal is to achieve acceleration with a multicore machine. Due to its black-box nature, the objective function may not be thread-safe, and therefore we have to use the process-level parallelization where the data is distributed to multiple processes. Sensor selection [13] and high-dimensional cox regression [14] are representative

Xiaoyu He, Zibin Zheng, Chuan Chen, and Yuren Zhou are with the School of Computer Science and Engineering, Sun Yat-sen University, Guangzhou 510006, P. R. China. Chuan Luo and Qingwei Lin are with Microsoft Research, P. R. China. Xiaoyu He is also with the School of Computer Science and Engineering, Nanyang Technological University, Singapore 639798. (E-mail: hxykokok@foxmail.com (X. He), zhzibin@mail.sysu.edu.cn (Z. Zheng), chenchuan@mail.sysu.edu.cn (C. Chen), zhousyuren@mail.sysu.edu.cn (Y. Zhou), chuan.luo@microsoft.com (C. Luo), qlin@microsoft.com (Q. Lin))

\* Corresponding Author: Z. Zheng

examples that are suitable for this scenario. These problems are in fact white-box, but the gradient evaluation is much more expensive than the function evaluation; treating them as black-box would heavily reduce the demand on computational resources. Generally, distributed black-box optimization offers a powerful search paradigm when calculating the gradient is expensive or infeasible, and it also retains the advantages from classical distributed computing frameworks in handling big data.

Black-box optimization methods, sometimes known as zeroth-order or derivative-free optimization methods, require only the availability of objective function values. They were among the earliest optimization methods in the history, while having attracted renewed interest recently due to the ubiquity of black-box models. The community has made several efforts in bringing the simplicity and universality of black-box optimization methods to the distributed world. A cornerstone of this research line is the Gaussian smoothing technique [15], a randomized finite-difference method that admits building a smooth surrogate of the original objective function with only zeroth-order information. With Gaussian smoothing, we can get a computationally cheap gradient estimator in the black-box setting, thereby making it possible to reuse existing first-order methods. Various black-box distributed optimization (DBO) methods have been proposed, based on the idea of hybridizing Gaussian smoothing with established distributed optimization methods [16], [17], [18], [19], [20], [21], [22]. These methods are typically easy to implement: the only work to do is to replace the real gradient with the one produced by Gaussian smoothing. The disadvantage is that they suffer a dimension-dependent slowdown in convergence rate, which is the cost must be paid for the absence of gradient information [23].

Current development in DBO methods has not yet been entirely successful; several common issues can be identified and should be carefully addressed. The first issue is the introduction of the smoothing parameter, which keeps a trade-off in improving the gradient estimation accuracy while avoiding roundoff errors [24, Chapter 8]. Tuning the smoothing parameter is onerous, and could become even more tricky in the distributed setting. This is caused by that its optimal value depends on the computing environment but different workers may have different environment. The second issue is the lack of adaptivity, in the sense that decision makers have to tune the step-sizes in workers or in the server or at both sides. Existing step-size adaptation rules cannot be generalized to black-box settings easily, as the gradient estimators produced by Gaussian smoothing does not meet the usual assumptions designed for first-order methods. There also exist algorithm-specific issues. For example, in [20], the authors found a well-developed distributed algorithm, signSGD [25], may fail to approach the optimality when extended to black-box settings, probably because of the unfordable sampling effort required for reducing the bias caused by Gaussian smoothing. For the above reasons, schemes that are based on Gaussian smoothing do not provide a truly seamless transformation of first-order distributed methods to the black-box setting. This calls for the need in developing new DBO methods based on completely different frameworks.

We propose in this work a new DBO method based on

evolution strategies (ESs) [26], [27], [28], a popular family of nature-inspired methods that excel in black-box real-valued optimization. Unlike Gaussian smoothing, ESs do not try to approximate the gradient or its surrogate, but instead guide the search with a probability distribution and gradually update this distribution on the fly. Moreover, the update of distribution is adaptive, requiring no knowledge about the landscape characteristics and very less user-supplied parameters. These features make ESs a strong candidate in designing new DBO methods and seem promising in addressing the aforementioned issues involved in Gaussian smoothing. ESs also possess a useful feature that they only use the comparison results of the objective function values among solutions, rather than their exact values [29]. This is likely to improve the robustness in the presence of noise, as the noise would not matter unless it changes the comparison results [30]. On the other hand, ESs are originally designed for non-distributed noise-less optimization and have not been extended to distributed settings. In fact, the major components of ESs are grounded on heuristics and a rigorous convergence analysis is still missing when applied on problems like (1). The goal of this paper is to help bridge this gap by describing ideas that can improve the applicability and rigorousness of ESs in the distributed stochastic setting. In particular, we propose a distributed evolution strategy (DES) with characteristics highlighted below:

- DES adopts a synchronous architecture that employs ESs to perform worker-side local updates and allows delayed averaging of individual decision vectors to reduce the communication overhead. It also supports server-side momentum, which is found to improve the performance in practice.
- When the local ES update is driven by an isotropic Gaussian distribution and when the function landscape is nonconvex, DES is competitive with existing zeroth-order methods in terms of iteration complexity. When certain sparsity assumption is met, DES can even align with the convergence rate of first-order methods.
- DES is fully adaptive in the sense that its convergence is guaranteed with any initial settings. Moreover, no numerical difference is involved so users will not worry about the roundoff errors.
- We propose two alternative probability distributions for generating mutation vectors in local updates. This significantly reduces the computation cost in high-dimensional settings.

In the remainder of this article, we first describe some related work in Section 2. In Section 3 we describe the details of DES and analyze its convergence properties. We then provide in Section 4 two alternative sampling methods and discuss their impact on the algorithm performance. Section 5 uses simulation studies to investigate the performance of our proposals. The article is concluded in Section 6. This paper has a supplement containing all proofs of our theoretical findings, as well as additional experimental results.

**Notation** Vectors are written in bold lowercase. We use  $\|\mathbf{x}\|_p$  to denote the  $\ell_p$  norm of  $\mathbf{x}$ . In addition, we use  $\|\mathbf{x}\|$  to denote a generic vector norm and  $\|\mathbf{x}\|_*$  its dual norm. We use  $\mathbb{E}[\cdot]$  to denote the expectation,  $\mathbb{V}[\cdot]$  the variance,

$\mathbb{I}\{\cdot\}$  the indicator function, and  $\mathbb{P}\{\cdot\}$  the probability. We use  $\mathbf{0}$  and  $\mathbf{I}$  to denote respectively the zero vector and the identity matrix of appropriate dimensions.  $\mathcal{N}(\mathbf{0}, \mathbf{I})$  denotes the multivariate isotropic Gaussian distribution. We use  $e_i$  to denote the  $i$ -th column of  $\mathbf{I}$ , i.e., the vector with 1 at the  $i$ -th coordinate and 0s elsewhere.

## 2 RELATED WORK

Distributed optimization is an active field in the optimization and machine learning communities. Our proposal belongs to the class of synchronous distributed optimization methods, which has been extensively studied in the first-order setting. Representative works include [31], [32], [33] and they are all based on the federated averaging (FedAvg) framework [34]. These methods usually use stochastic gradient descent (SGD) as the worker-side solver while adopting different schemes to improve communication efficiency. Theoretically, these first-order methods could be generalized straightforwardly to the black-box setting via Gaussian smoothing; but to the best of our knowledge, heretofore there exist no generic zeroth-order approaches in the synchronous distributed setting. The closest work is the zeroth-order version of signSGD, ZO-signSGD, described in [20]; however, this method requires communication per iteration and does not guarantee global convergence. Another relevant method is FedProx [35] which does not rely on the specification of local solvers. FedProx technically admits using zeroth-order solvers at the worker-side, but it requires an additional regularization parameter to guarantee local functions becoming strongly convex; in this sense, it is not applicable when the function is black-box. On the other hand, there exist several DBO methods built on asynchronous parallel [19], [36] or multi-agent architectures [18], [21]; but they are not applicable in the synchronous distributed setting which is the main focus of this work.

Choosing the step-size is critical in implementing stochastic optimization methods, as one cannot simply use a line search when the landscape is noisy. To avoid the tedious step-size tuning phase, a variety of adaptation schemes have been proposed for first-order stochastic methods, where the step-size is updated with historical first-order information. Remarkable examples includes [37], [38], [39], [40]. However, only a few of these adaptation schemes have been extended to the distributed setting, e.g., in [41], [42], [43], and they still require a manually selected step-size for each worker. The method proposed in this work, on the contrary, can automatically choose step-sizes for both the server and the workers, and seems to be the first one that achieves such “full adaptivity”.

Moving beyond the classical approaches that are based on rigorous mathematic tools, studies on stochastic optimization are very scarce in the evolutionary computation community. Almost all existing studies consider a more generic setting, the noisy optimization, and do not explore the expectation structure of problem (1); see [44] for a survey. It is found in [45], [46] that, via simple resampling, modern ESs originally designed for deterministic optimization may achieve the best known convergence rate on noisy landscapes [47]. These studies, however, require assumptions

that are completely different from the ones used in classical literatures. It is still unknown how evolutionary algorithms perform on problem (1) with more generic assumptions.

Various studies on distributed optimization exist in the evolutionary community; related methodologies and tools have been nicely summarized in [48], [49]. As evolutionary approaches are usually population-based, these studies mostly focus on the parallel acceleration of the function evaluations of population, but have seldom touched the data decentralization (which is the case of this study). In this study, the distributed framework is mainly designed to achieve data decentralization; but parallelization is also supported in a synchronous manner.

## 3 THE PROPOSED METHOD: DES

In this section, we first propose a modified ES method for non-distributed deterministic optimization and then use it as a building block to develop the DES algorithm. Although this work focus on black-box optimization, we need the following assumptions to analyze the performance of DES. Unless stated otherwise, we assume  $\mathbb{R}^n$  is equipped with some generic vector norm  $\|\cdot\|$  and its dual norm is denoted by  $\|\cdot\|_*$ .

**Assumption 1.** *The function  $F$  has Lipschitz continuous gradient with constant  $L$  for any  $\xi$ , i.e.,*

$$\|\nabla F(\mathbf{x}; \xi) - \nabla F(\mathbf{y}; \xi)\|_* \leq L \|\mathbf{x} - \mathbf{y}\| \quad \forall \mathbf{x}, \mathbf{y} \in \mathbb{R}^n.$$

**Assumption 2.** *The gradient of  $F$  has bounded variance, i.e.,*

$$\mathbb{E} [\|\nabla F(\mathbf{x}; \xi) - \nabla f(\mathbf{x})\|_*^2] \leq \sigma^2 \quad \forall \mathbf{x} \in \mathbb{R}^n.$$

**Assumption 3.** *Every worker has access to the distribution of  $\xi$  independently and identically.*

Assumptions 1 and 2 are customary in the analysis of stochastic optimization. They are useful when using gradients in measuring the optimality on nonconvex landscapes. Assumption 3 is somewhat restrictive; but it is required to reduce the global variance via minibatching at the worker-side. On the other hand, as an adaptive method, our method does not assume the gradients to be universally bounded, and this is an advantage over several existing methods (e.g., [37], [38]).

### 3.1 A modified ES for deterministic optimization

We first consider the simplest ES framework, usually termed as  $(1 + 1)$ -ES in the literature, where in each iteration a parent produces a single offspring using mutation and the one with a better objective value becomes the new parent. The mutation is typically performed with an isotropic Gaussian perturbation and its variance is gradually updated. The pseudo-code of this method is given in Algorithm 1. Specifically, it maintains a vector  $\mathbf{x}_k \in \mathbb{R}^n$  to encode the parent solution and a scalar  $\alpha_k$  the standard variance. The vector  $\mathbf{u}_k \in \mathbb{R}^n$  (called mutation vector) is drawn from the standard Gaussian distribution and then used to construct the offspring given by  $\mathbf{x}_k + \alpha_k \mathbf{u}_k$ . Hereinafter we call  $\alpha_k$  the step-size because it (approximately) determines the length of the descent step.

---

**Algorithm 1** A modified ES implementation for deterministic nonconvex optimization
 

---

**Require:**  $\mathbf{x}_0 \in \mathbb{R}^n$ : initial solution;  $\alpha_0 \in \mathbb{R}_+$ : initial step-size

- 1: **for**  $k = 0, 1, \dots, K - 1$  **do**
- 2:    $\alpha_k = \alpha_0 / \sqrt{k + 1}$
- 3:   Sample  $\mathbf{u}_k$  from  $\mathcal{N}(\mathbf{0}, \mathbf{I})$
- 4:   **if**  $f(\mathbf{x}_k + \alpha_k \mathbf{u}_k) \leq f(\mathbf{x}_k)$  **then**
- 5:      $\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{u}_k$
- 6:   **else**
- 7:      $\mathbf{x}_{k+1} = \mathbf{x}_k$
- 8:   **end if**
- 9: **end for**

---

The only difference of our implementation to existing ones lies in the specification of step-sizes: here we use a pre-defined diminishing rule (in Line 2) while almost all modern ESs adopt a comparison-based adaptation rule. Precisely, most ESs obtain  $\alpha_{k+1}$  via multiplying  $\alpha_k$  by some factor that depends on whether the offspring is better than the parent. This admits the step-size to shrink exponentially fast, so ESs may achieve linear convergence on certain landscapes [50]. In this work, however, the objective landscape is generally nonconvex, so we cannot expect more than sublinear convergence [51]. It suggests a thorough redesign of the step-size rule.

Our choice of the step-size rule  $\alpha_k = \alpha_0 / \sqrt{k + 1}$  is to align with the known convergence rate on deterministic nonconvex functions,  $\mathcal{O}(1/K)$ , measured by the squared gradient norm. This is illustrated in the following theorem.

**Theorem 1.** *Let Assumption 1 hold with the self-dual  $\ell_2$  norm, i.e.,  $\|\cdot\| = \|\cdot\|_* = \|\cdot\|_2$ . Assume the function  $f$  is bounded below by  $f_*$ . The iterations generated by Algorithm 1 satisfy*

$$\begin{aligned} & \frac{1}{K} \sum_{k=0}^{K-1} \mathbb{E} [\|\nabla f(\mathbf{x}_k)\|_2] \\ & \leq \sqrt{\frac{2\pi}{K}} \left( \frac{f(\mathbf{x}_0) - f_*}{\alpha_0} + \alpha_0 \ln(1 + \log K) \right). \end{aligned} \quad (2)$$

Define  $\Delta_f = f(\mathbf{x}_0) - f_*$ . The bound in (2) is minimized at  $\alpha_0 = \Theta\left(\sqrt{\frac{\Delta_f}{\ln K}}\right)$ ; in this case, we have, via taking the square on both sides, the following rate for ES:

$$\left( \frac{1}{K} \sum_{k=0}^{K-1} \mathbb{E} [\|\nabla f(\mathbf{x}_k)\|_2] \right)^2 \leq \tilde{\mathcal{O}}\left(\frac{\Delta_f \ln K}{K}\right) \quad (3)$$

where  $\tilde{\mathcal{O}}$  hides the negligible  $\log K$  term in the  $\mathcal{O}$  notation. Whereas, for comparison, the best known bound for gradient descent is

$$\frac{1}{K} \sum_{k=0}^{K-1} \mathbb{E} [\|\nabla f(\mathbf{x}_k)\|_2^2] \leq \mathcal{O}\left(\frac{\Delta_f L}{K}\right), \quad (4)$$

or, if the gradient is estimated using Gaussian smoothing,

$$\frac{1}{K} \sum_{k=0}^{K-1} \mathbb{E} [\|\nabla f(\mathbf{x}_k)\|_2^2] \leq \mathcal{O}\left(\frac{\Delta_f \ln K}{K}\right). \quad (5)$$

See [52] for these results. These bounds are quite similar, except for the difference in measuring the optimality. It

suggests that 1) the proposed modified ES is competitive with zeroth-order gradient descent methods that are based on Gaussian smoothing, and 2) is only  $n$  times slower than first-order gradient descent methods. The slowdown compared to first-order methods is probably due to that the mutation in ES is not necessarily a descent step and it has a dimension-dependent variance. The advantage of ES is twofold: it does not need to estimate the gradient and it converges with any step-size setting.

### 3.2 Implementation of DES

We now describe the DES method for handling distributed stochastic problems. Algorithm 2 provides the pseudo-code for our method. DES adopts the well-known federated averaging framework and uses the deterministic ES proposed in Section 3.1 as worker-side solvers. Its search process is divided into  $T$  rounds, and in the  $t$ -th round, the server maintains a solution  $\mathbf{x}_t \in \mathbb{R}^n$ , a step-size  $\alpha_0^t \in \mathbb{R}_+$ , and an optional momentum term  $\mathbf{m}_t \in \mathbb{R}^n$ . The step-size should decrease at a  $1/T^{0.25}$  rate to achieve convergence. The momentum term is to enhance the robustness of the server-side updates.

At the beginning of the  $t$ -th round, the server broadcasts  $\mathbf{x}_t$  and  $\alpha_0^t$  to all  $M$  workers, and the workers use them as their initial solutions and step-sizes respectively (in Lines 3-4). Each worker  $i$  then draws a minibatch  $\mathcal{D}_i$  of size  $b$  randomly<sup>1</sup> and constructs a stochastic approximated function  $f_i$  (in Lines 5-6). The minibatch  $\mathcal{D}_i$  is fixed during this round and thus the function  $f_i$  is considered as deterministic. The  $i$ -th worker then optimizes  $f_i$  using the deterministic ES with a budget of  $K$  iterations. At the  $k$ -th iteration of the  $i$ -th worker, we denote respectively the solution and step-size as  $\mathbf{v}_{i,k}^t$  and  $\sigma_k^t$ . After the worker-side search phase terminates, all workers upload their final output (i.e.,  $\mathbf{v}_{i,K}^t$ ), and then the server computes an averaged descent step, denote by  $\mathbf{d}_{t+1}$ , in Line 17. Before the end of the  $t$ -th round, as shown in Lines 18-19, the server accumulates the descent step into the momentum  $\mathbf{m}_{t+1}$ , with a parameter  $\beta$  controlling the rate, and finally obtains the new solution  $\mathbf{x}_{t+1}$  via moving  $\mathbf{x}_t$  along the momentum direction. Note that in the final step we do not specify a step-size; the magnitude of how the solution is updated is implicitly controlled by the deterministic ES at the worker-side. This is the critical step for achieving full adaptivity.

### 3.3 Convergence properties

We now analyze the convergence behavior of DES. Firstly we consider a general setting where the optimality is measured by the  $\ell_2$  norm of the gradient.

**Theorem 2.** *Let Assumptions 1 to 3 hold with the self-dual  $\ell_2$  norm, i.e.,  $\|\cdot\| = \|\cdot\|_* = \|\cdot\|_2$ . Assume the function  $f$  is*

1. To simplify the analysis, throughout this work, we assume the minibatch to be drawn uniformly with replacement.

**Algorithm 2 DES**


---

**Require:**  $\mathbf{x}_0 \in \mathbb{R}^n$ : initial solution;  $\alpha \in \mathbb{R}_+$ : initial step-size;  $\beta \in [0, \sqrt{\frac{1}{2\sqrt{2}}})$ : momentum parameter;  $b \geq \sqrt{T}$ : minibatch size

- 1: **for**  $t = 0, 1, \dots, T - 1$  **do**
- 2:   **for**  $i = 1, 2, \dots, M$  **in parallel do**
- 3:      $\mathbf{v}_{i,0}^t = \mathbf{x}_t$
- 4:      $\alpha_0^t = \alpha/(t+1)^{0.25}$
- 5:     Draw a minibatch  $\mathcal{D}_i$  of size  $b$
- 6:     Define  $f_i(\mathbf{x}) = \frac{1}{b} \sum_{\xi \in \mathcal{D}_i} F(\mathbf{x}; \xi)$
- 7:     **for**  $k = 0, 1, \dots, K-1$  **do**
- 8:        $\alpha_k^t = \alpha_0^t/(k+1)^{0.5}$
- 9:       Sample  $\mathbf{u}_{i,k}^t$  from  $\mathcal{N}(\mathbf{0}, \mathbf{I})$
- 10:       **if**  $f_i(\mathbf{v}_{i,k}^t + \alpha_k^t \mathbf{u}_{i,k}^t) \leq f_i(\mathbf{v}_{i,k}^t)$  **then**
- 11:          $\mathbf{v}_{i,k+1}^t = \mathbf{v}_{i,k}^t + \alpha_k^t \mathbf{u}_{i,k}^t$
- 12:       **else**
- 13:          $\mathbf{v}_{i,k+1}^t = \mathbf{v}_{i,k}^t$
- 14:       **end if**
- 15:     **end for**
- 16:   **end for**
- 17:    $\mathbf{d}_{t+1} = \frac{1}{M} \sum_{i=1}^M \mathbf{v}_{i,K}^t - \mathbf{x}_t$
- 18:    $\mathbf{m}_{t+1} = \beta \mathbf{m}_t + (1-\beta) \mathbf{d}_{t+1}$
- 19:    $\mathbf{x}_{t+1} = \mathbf{x}_t + \mathbf{m}_{t+1}$
- 20: **end for**

---

bounded below by  $f_*$  and choose  $0 \leq \beta < \sqrt{\frac{1}{2\sqrt{2}}}, b \geq \sqrt{T}$ . The iterations generated by Algorithm 2 satisfy

$$\begin{aligned} \frac{1}{T} \sum_{t=0}^{T-1} \mathbb{E}[\|\nabla f(\mathbf{x}_t)\|_2] &\leq \frac{\sqrt{2\pi}}{T^{3/4}} \frac{f(\mathbf{x}_0) - f_*}{\alpha\sqrt{K}} \\ &+ \frac{\sqrt{n}}{T^{1/4}} \left( 2\alpha L \left( \sqrt{2\pi n} \Psi + \frac{80\beta\sqrt{K}}{3} \right) + \frac{8\sqrt{2\pi}\sigma}{3} \right) \quad (6) \end{aligned}$$

where

$$\Psi = \left( \left( \frac{2}{1-2\sqrt{2}\beta^2} + \frac{1}{2} \right) \sqrt{K} + \frac{1}{2\sqrt{K}} \right) (1 + \log K) + \sqrt{K} \quad (7)$$

Here we briefly discuss our theoretical result and its implications.

**Remark** (Convergence rate). When  $K$  is fixed, the DES method achieves  $\mathcal{O}(T^{-1/4})$  rate in terms of  $\frac{1}{T} \sum_{t=0}^{T-1} \mathbb{E}[\|\nabla f(\mathbf{x}_t)\|_2]$ . If, in addition, setting  $\alpha = \Theta(n^{-1/2}L^{-1})$ , we achieve

$$\left( \frac{1}{T} \sum_{t=0}^{T-1} \mathbb{E}[\|\nabla f(\mathbf{x}_t)\|_2] \right)^2 \leq \mathcal{O}\left(\sigma^2 \frac{n}{\sqrt{T}}\right). \quad (8)$$

The dependence on  $T$  aligns with the best known bound for zeroth-order stochastic methods, e.g., in [52], which can be rewritten as

$$\frac{1}{T} \sum_{t=0}^{T-1} \mathbb{E}[\|\nabla f(\mathbf{x}_t)\|_2^2] \leq \mathcal{O}\left(\sigma \sqrt{\frac{\Delta_f L n}{T}}\right) \quad (9)$$

where  $\Delta_f = f(\mathbf{x}_0) - f_*$ . Our method has a worse dependence on  $\sigma$ . However, the best known bound in (9) requires  $\sigma$  to be known when setting the step-size; so it remains unknown whether the dependence of  $\sigma$  is improvable in a

real black-box setting. Our obtained rate (8), in fact, matches the rate of adaptive gradient methods [41] in terms of the  $\sigma$ -dependence. The convergence of DES is less dependent on the function landscape characteristics (e.g.,  $\Delta_f$  and  $L$ ), at the cost of having a worse dimension-dependence. This indicates that DES might suffer from the curse of dimensionality but could be better in handling ill-conditioning and robust to initialization.

**Remark** (Minibatching). The setting  $b \geq \sqrt{T}$  is critical in achieving convergence. This requirement is not usual for first-order methods or Gaussian smoothing based zeroth-order methods, since for these methods the gradient variance can be scaled down by choosing a sufficiently small step-size. The DES method only relies on the comparison results among solutions and does not try to estimate the gradient, so the bias of the descent step could accumulate and prevent convergence unless a large minibatch is used to explicitly reduce the noise. Note that similar issues are encountered in the signSGD method [25] where the descent step becomes biased due to the sign operation. signSGD, however, requires  $b \geq T$  to achieve convergence whereas in our method it is relaxed to  $b \geq \sqrt{T}$ .

**Remark** (Adaptivity). The DES method is fully adaptive in the sense that it converges with any valid parameter setting and relies no knowledge about landscape characteristics (e.g., values of  $L$  and  $\sigma$ ). In contrast to existing distributed adaptive gradient methods such as [41], [42], DES does not need the gradient to be uniformly bounded and does not involve a non-adaptive worker-side step-size.

**Remark** (Momentum). The bound in (6) suggests that the optimal  $\beta$  is 0, but in experiments we found choosing  $\beta > 0$  in most cases leads to better performance. This is probably because the suggested rate is overestimated, so it does not reflect how the momentum influences the algorithm performance. The impact of this parameter will be investigated using simulation studies.

It is found from (8) that the DES method suffers a dimension-dependence slowdown in convergence. We note, however, that when the landscape exhibits certain sparse structure, DES may automatically exploit such sparsity and achieve speedup. This is formally stated below:

**Theorem 3.** Let Assumptions 1 to 3 hold with the  $\ell_\infty$  norm, i.e.,  $\|\cdot\| = \|\cdot\|_\infty$  and  $\|\cdot\|_* = \|\cdot\|_1$ . Assume the function  $f$  is bounded below by  $f_*$  and choose  $0 \leq \beta < \sqrt{\frac{1}{2\sqrt{2}}}, b \geq \sqrt{T}$ . If  $\|\nabla f(\mathbf{x})\|_0 \leq s$  for any  $\mathbf{x} \in \mathbb{R}^n$  and some constant  $s \leq n$ , then the iterations generated by Algorithm 2 satisfy

$$\begin{aligned} \frac{1}{T} \sum_{t=0}^{T-1} \mathbb{E}[\|\nabla f(\mathbf{x}_t)\|_1] &\leq \frac{\sqrt{2\pi s}}{T^{3/4}} \frac{f(\mathbf{x}_0) - f_*}{\alpha\sqrt{K}} \\ &+ \frac{8\sqrt{\log(\sqrt{2n})}}{T^{1/4}} \left\{ \alpha L \left( \sqrt{2\pi s \log(\sqrt{2n})} \Psi + \frac{10\beta\sqrt{K}}{3} \right) \right. \\ &\left. + \frac{2\sqrt{2\pi s\sigma}}{3} \right\} \quad (10) \end{aligned}$$

where  $\Psi$  is defined in (7).

**Remark** (Adaptation to sparsity). The rate established above only poly-logarithmically depends on the dimension. With any setting of  $\alpha$  and noting the fact  $\|\nabla f(\mathbf{x})\|_1 \geq \|\nabla f(\mathbf{x})\|_2$ , we have

$$\left( \frac{1}{T} \sum_{t=0}^{T-1} \mathbb{E}[\|\nabla f(\mathbf{x}_t)\|_2] \right)^2 \leq \tilde{\mathcal{O}}\left(\frac{\sigma^2}{\sqrt{T}}\right)$$

which is nearly independent of the dimension, as in most first-order methods. We emphasize the improvement in the dimension-dependence is achieved automatically when the landscape is sparse, without any modification made to the algorithm.

## 4 ALTERNATIVE SAMPLING SCHEMES

One bottleneck of the DES method implemented in Section 3 is the generation of mutation vectors, in which a huge amount of Gaussian random numbers are required. It is known that generating Gaussian random numbers is usually expensive, and it may cause efficiency issue in high-dimensional settings. In this section we propose two alternative probability models which can be used in DES for improving the sampling efficiency.

### 4.1 Mixture sampling for fast mutation

In Algorithm 2, each worker has to perturb its maintained solution in all coordinates, leading to the  $O(n)$  complexity per-iteration. Our scheme to improve this is to only perturb a small subset of the coordinates. Specifically, at each worker's iteration we uniformly and randomly sample a subset of  $l$  coordinates with replacement, where  $l \ll n$  is a small integer. Then, on each selected coordinate, we perturb the current solution with a univariate random noise. This two-level sampling strategy yields a mixture distribution since its samples follow a mixture of  $n$  univariate probability models defined individually on each coordinate. Statistical characteristics of this mixture distribution is completely determined by the parameter  $l$  and the underlying univariate model. In the following we provide two ways in designing the mixture sampling scheme.

The first scheme is to use Gaussian distribution on each selected coordinate and we call it “*mixture Gaussian sampling*”. This scheme works via replacing the Gaussian distribution (e.g.,  $\mathcal{N}(\mathbf{0}, \mathbf{I})$  in Algorithm 2) with the probability model defined below:

**Definition 1.** We call a random vector  $\mathbf{u} \in \mathbb{R}^n$  is obtained from the mixture Gaussian sampling if it can be expressed as

$$\mathbf{u} = \sqrt{\frac{n}{l}} \sum_{j=1}^l e_{r_j} z_j$$

where  $z_1, \dots, z_l$  are scalars drawn independently from  $\mathcal{N}(0, 1)$ , and  $r_1, \dots, r_l$  are integers drawn uniformly from  $\{1, \dots, n\}$  with replacement. We denote its underlying probability model by  $\mathcal{M}_l^G$ .

The second scheme is to use, on each selected coordinate, the Rademacher distribution which belongs to the sub-Gaussian family. We call this scheme “*mixture Rademacher*

---

### Algorithm 3 DES with mixture sampling

---

**Require:**  $\mathbf{x}_0 \in \mathbb{R}^n$ : initial solution;  $\alpha \in \mathbb{R}_+$ : initial step-size;  $\beta \in [0, \sqrt{\frac{1}{2\sqrt{2}}}]$ : momentum parameter;  $b \geq \sqrt{T}$ : minibatch size;  $l \in \mathbb{Z}_+$ : mixture parameter

- 1: **for**  $t = 0, 1, \dots, T - 1$  **do**
- 2:   **for**  $i = 1, 2, \dots, M$  in parallel **do**
- 3:      $\mathbf{v}_{i,0}^t = \mathbf{x}_t$
- 4:      $\alpha_0^t = \alpha/(t+1)^{0.25}$
- 5:     Draw a minibatch  $\mathcal{D}_i$  of size  $b$
- 6:     Define  $f_i(\mathbf{x}) = \frac{1}{b} \sum_{\xi \in \mathcal{D}_i} F(\mathbf{x}; \xi)$
- 7:     **for**  $k = 0, 1, \dots, K-1$  **do**
- 8:        $\alpha_k^t = \alpha_0^t/(k+1)^{0.5}$
- 9:        $\mathbf{w} = \mathbf{v}_{i,k}^t$
- 10:       **for**  $j = 1, \dots, l$  **do**
- 11:          Draw  $r$  randomly uniformly from  $\{1, \dots, n\}$  with replacement
- 12:          Option I (mixture Gaussian sampling):  
             $z \sim \mathcal{N}(0, 1)$
- 13:          Option II (mixture Rademacher sampling):  
             $z$  is either -1 or 1 with 50% chance
- 14:           $w_r = w_r + \alpha_k^t \sqrt{\frac{n}{l}} z$
- 15:        **end for**
- 16:        **if**  $f_i(\mathbf{w}) \leq f_i(\mathbf{v}_{i,k}^t)$  **then**
- 17:           $\mathbf{v}_{i,k+1}^t = \mathbf{w}$
- 18:        **else**
- 19:           $\mathbf{v}_{i,k+1}^t = \mathbf{v}_{i,k}^t$
- 20:        **end if**
- 21:       **end for**
- 22:     **end for**
- 23:      $\mathbf{d}_{t+1} = \frac{1}{M} \sum_{i=1}^M \mathbf{v}_{i,K}^t - \mathbf{x}_t$
- 24:      $\mathbf{m}_{t+1} = \beta \mathbf{m}_t + (1 - \beta) \mathbf{d}_{t+1}$
- 25:      $\mathbf{x}_{t+1} = \mathbf{x}_t + \mathbf{m}_{t+1}$
- 26: **end for**

---

sampling”. In this case, the mutation vector is drawn from the following distribution:

**Definition 2.** We call a random vector  $\mathbf{u} \in \mathbb{R}^n$  is obtained from the mixture Rademacher sampling if it can be expressed as

$$\mathbf{u} = \sqrt{\frac{n}{l}} \sum_{j=1}^l e_{r_j} z_j$$

where  $z_1, \dots, z_l$  are independent scalars to be either 1 or -1 with 50% chance, and  $r_1, \dots, r_l$  are integers drawn uniformly from  $\{1, \dots, n\}$  with replacement. We denote its underlying probability model by  $\mathcal{M}_l^R$ .

The coefficient  $\sqrt{\frac{n}{l}}$  in the above definitions is to normalize the probability model to achieve the identity covariance matrix, which will be illustrated in the subsequent analyses. When  $l \leq n$ , we can implement the above sampling schemes efficiently in DES, via a loop of length  $l$  applied on the solutions maintained at the worker-side. Algorithm 3 gives the detailed implementations of this idea. When  $l \ll n$ , the time complexity for sampling can be reduced to  $O(l)$ , and this will save the computing time considerably when  $n$  is large.

### 4.2 Behavior of DES with mixture sampling

We first discuss the statistic characteristics of proposed two sampling schemes.

The mixture Gaussian sampling, in the case of  $l \rightarrow \infty$ , will degenerate to the standard Gaussian sampling. This limiting case, to some extent, is useless as it will make the sampling even more expensive. Therefore, we are more interested in the  $l \ll n$  case. The following describes the statistical properties that are required in understanding the mixture sampling schemes. Since the probability model is symmetric by design, we will focus on its second-order and fourth-order moments.

**Proposition 1.** Let  $l \in \mathbb{Z}_+$  and  $\mathbf{u} \in \mathbb{R}^n$ . If  $\mathbf{u} \sim \mathcal{M}_l^G$ , we have  $\mathbb{V}[\mathbf{u}] = \mathbf{I}$  and

$$\mathbb{E}[|\mathbf{y}^T \mathbf{u}|^4] = 3 \left( \frac{n}{l} \|\mathbf{y}\|_4^4 + \frac{l-1}{l} \|\mathbf{y}\|_2^4 \right), \quad \forall \mathbf{y} \in \mathbb{R}^n. \quad (11)$$

The above shows that the mixture Gaussian sampling will generate mutation vectors having exactly the same covariance matrix as the standard Gaussian sampling, regardless of the  $l$  value. In addition, since  $n\|\mathbf{y}\|_4^4 \geq \|\mathbf{y}\|_2^4 \geq \|\mathbf{y}\|_4^4$ , we know

$$\frac{\mathbb{E}[|\mathbf{y}^T \mathbf{u}|^4]}{\|\mathbf{y}\|_2^4} \in \left[ 3, 3 \frac{n+l-1}{l} \right],$$

which then indicates that any 1-dimensional projection of  $\mathcal{M}_l^G$  will have a larger kurtosis than Gaussian. Implications of this property are twofold. Firstly, the mixture Gaussian sampling method is more likely to generate outliers in the mutation phase, so if the landscape is highly multimodal, DES equipped with  $\mathcal{M}_l^G$  would have a greater chance to escape local optima. Secondly, this makes DES prefer exploration than exploitation, and hence, it may degrade the performance. We note, as will be demonstrated later, that such a performance degradation is insignificant when the gradient is dense.

Similarly, the mixture Rademacher sampling can be characterized as below.

**Proposition 2.** Let  $l \in \mathbb{Z}_+$  and  $\mathbf{u} \in \mathbb{R}^n$ . If  $\mathbf{u} \sim \mathcal{M}_l^R$ , we have  $\mathbb{V}[\mathbf{u}] = \mathbf{I}$  and

$$\mathbb{E}[|\mathbf{y}^T \mathbf{u}|^4] = \frac{n}{l} \|\mathbf{y}\|_4^4 + 3 \frac{l-1}{l} \|\mathbf{y}\|_2^4, \quad \forall \mathbf{y} \in \mathbb{R}^n. \quad (12)$$

Again, the mixture Rademacher sampling is more likely to produce outlier mutation vectors than the standard Gaussian sampling, while they have the same covariance matrix. But it is found, via comparing (12) with (11), that  $\mathcal{M}_l^R$  can scale down the kurtosis of  $\mathcal{M}_l^G$  by a factor about  $1/3$  for sufficiently large  $n$ . In this sense, the mixture Rademacher sampling can be considered as a trade-off between the standard Gaussian sampling and the mixture Gaussian sampling.

In the following, we analyze the convergence performance of DES when equipped with the mixture sampling schemes. For expository purposes, we assume  $\beta = 0$  and only consider the  $\ell_2$  norm case, though our analysis can be extended directly to a more general setting.

**Theorem 4.** Let Assumptions 1 to 3 hold with the self-dual  $\ell_2$  norm, i.e.,  $\|\cdot\| = \|\cdot\|_* = \|\cdot\|_2$ . Assume the function  $f$  is bounded below by  $f_*$  and choose  $\beta = 0, b \geq \sqrt{T}$ . If  $\|\nabla f(\mathbf{x})\|_2^4/\|\nabla f(\mathbf{x})\|_4^4 \geq \tilde{s}$  for any  $\mathbf{x} \in \mathbb{R}^n$  and some constant  $\tilde{s} \in [1, n]$ , then the iterations generated by Algorithm 3 with mixture Gaussian sampling satisfy

$$\begin{aligned} \frac{1}{T} \sum_{t=0}^{T-1} \mathbb{E}[\|\nabla f(\mathbf{x}_t)\|_2] &\leq \sqrt{3 + \frac{3n}{\tilde{s}l}} \left\{ \frac{2}{T^{3/4}} \frac{f(\mathbf{x}_0) - f_*}{\alpha\sqrt{K}} \right. \\ &\quad \left. + \frac{4\sqrt{n}}{T^{1/4}} \left( \frac{4}{3}\sigma + L\sqrt{n}\hat{\Psi}\alpha \right) \right\} \end{aligned} \quad (13)$$

where

$$\hat{\Psi} = \left( \frac{1}{2\sqrt{K}} + \frac{5}{2}\sqrt{K} \right) (1 + \log K) + \frac{1}{\sqrt{K}}.$$

**Remark** (Impact of the denseness). The bound in (13) is generally looser than that for DES with standard Gaussian sampling. For example, consider setting  $\alpha = \Theta(n^{-1/2}L^{-1})$ , then we obtain the convergence rate

$$\left( \frac{1}{T} \sum_{t=0}^T \mathbb{E}[\|\nabla f(\mathbf{x}_t)\|_2] \right)^2 \leq \mathcal{O} \left( \frac{\sigma^2 n^2}{\tilde{s}l\sqrt{T}} \right),$$

which could be  $\mathcal{O}(\frac{n}{\tilde{s}l})$  times slower than the rate given in (8). The involved constant  $\tilde{s}$ , by the definition of vector norms, always exists in the range  $[1, n]$ . In fact, as has been pointed out in [53], the quantity  $\|\mathbf{y}\|_4^4/\|\mathbf{y}\|_2^4$  measures the sparseness of a vector  $\mathbf{y} \in \mathbb{R}^n$ , so the constant  $\tilde{s}$  here can be viewed as a lower bound of the denseness of the gradient  $\nabla f(\mathbf{x})$ . If the gradient is relatively dense, e.g., all coordinates in the gradient are of a similar magnitude, then  $\tilde{s}$  will be close to  $n$ . In this case, the convergence rate with mixture Gaussian sampling will coincide with that with standard Gaussian sampling. We may therefore conclude, by comparing Theorems 3 and 4, that the mixture sampling is more suitable for dense problems whereas the standard Gaussian sampling is preferred for sparse problems.

**Theorem 5.** Let Assumptions 1 to 3 hold with the self-dual  $\ell_2$  norm, i.e.,  $\|\cdot\| = \|\cdot\|_* = \|\cdot\|_2$ . Assume the function  $f$  is bounded below by  $f_*$  and choose  $\beta = 0, b \geq \sqrt{T}$ . If  $\|\nabla f(\mathbf{x})\|_2^4/\|\nabla f(\mathbf{x})\|_4^4 \geq \tilde{s}$  for any  $\mathbf{x} \in \mathbb{R}^n$  and some constant  $\tilde{s} \in [1, n]$ , then the iterations generated by Algorithm 3 with mixture Rademacher sampling satisfy

$$\begin{aligned} \frac{1}{T} \sum_{t=0}^{T-1} \mathbb{E}[\|\nabla f(\mathbf{x}_t)\|_2] &\leq \sqrt{3 + \frac{n}{\tilde{s}l}} \left\{ \frac{2}{T^{3/4}} \frac{f(\mathbf{x}_0) - f_*}{\alpha\sqrt{K}} \right. \\ &\quad \left. + \frac{4\sqrt{n}}{T^{1/4}} \left( \frac{4}{3}\sigma + L\sqrt{n}\hat{\Psi}\alpha \right) \right\} \end{aligned} \quad (14)$$

where  $\hat{\Psi}$  is defined as in Theorem 4.

The bound corresponding to the mixture Rademacher sampling is slightly tighter than that for the mixture Gaussian sampling. This could make a considerable difference in practice when  $n$  is large. Our empirical studies show that in certain cases the mixture Rademacher sampling could be better than the mixture Gaussian sampling, while their performance is in general similar.

## 5 SIMULATION STUDY

In this section we perform simulations to investigate the empirical performance of the proposed methods.

## 5.1 Experimental settings

We consider three binary classification problems arising in machine learning and statistics. They include logistic regression (LR), nonconvex support vector machine (NSVM), and linear support vector machine (LSVM) with a hinge loss. For these problems, the random sample  $\xi$  corresponds to a pair of input vector  $\mathbf{z}$  and target label  $y$ , and the objective function takes the finite-sum form:

$$f(\mathbf{x}) = \frac{1}{N} \sum_{i=1}^N F(\mathbf{x}; \xi_i) = \frac{1}{N} \sum_{i=1}^N \text{loss}(\mathbf{x}; \mathbf{z}_i, y_i) + \frac{\lambda_p}{2} \|\mathbf{x}\|_2^2,$$

where  $\text{loss}$  is the loss function and  $\lambda_p$  is the regularization parameter. We fix  $\lambda_p = 10^{-6}$  throughout this study. The loss function is defined as

- Logistic Regression (LR)

$$\text{loss}(\mathbf{x}; \mathbf{z}, y) = \log(1 + \exp(-y(\mathbf{x}^T \mathbf{z})))$$

- Nonconvex Support Vector Machine (NSVM)

$$\text{loss}(\mathbf{x}; \mathbf{z}, y) = 1 - \tanh(y(\mathbf{x}^T \mathbf{z}))$$

- Linear Support Vector Machine (LSVM)

$$\text{loss}(\mathbf{x}; \mathbf{z}, y) = \max \left\{ 0, 1 - y(\mathbf{x}^T \mathbf{z}) \right\}.$$

LR is the simplest, being strongly convex and smooth. NSVM is nonconvex but smooth. LSVM is not smooth so it does not meet our assumption; we choose it to verify the robustness of our proposals.

Six datasets widely used for benchmarking stochastic optimization methods are selected and their properties are briefly summarized in Table 1. For each dataset, 80% data are chosen for training and the remaining 20% are for testing. We partition the training samples uniformly into  $M$  pieces with no overlap, and each piece is stored at a counterpart worker.

TABLE 1  
Statistics of the used datasets.

dataset	$n$	$N$
ijcnn1	22	49990
SUSY	18	5000000
covtype	54	581012
mnist	780	60000
real-sim	20958	72309
rcv1	47236	677399

We implement four algorithms for comparison, including federated zeroth-order gradient method (Fed-ZO-GD), federated zeroth-order SGD (Fed-ZO-SGD), zeroth-order signSGD method (ZO-signSGD), and the standard ES with cumulative step-size adaptation (ES-CSA). Fed-ZO-GD, Fed-ZO-SGD, and ZO-signSGD are distributed algorithms based on gradient estimation. ES-CSA is non-distributed but we have made some modifications to enable distributed optimization. Their configurations are described below:

1. All datasets are available at <https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets>. The mnist dataset is transformed into binary class based on whether the label (digital) is greater than 4.

- Fed-ZO-GD. It is implemented by replacing the worker-side solver of DES with the Gaussian smoothing based gradient descent method, so it can be considered as a plain combination of FedAvg and the zeroth-order gradient descent method. Each worker individually chooses a random minibatch of size  $b$  in each round and runs zeroth-order gradient descent for  $K' = K/2$  iterations with the step-size  $\alpha_k^t = \frac{\alpha_0^t}{k+1} = \frac{\alpha}{(k+1)\sqrt{t+1}}$ . We choose the central-difference in Gaussian smoothing, so each worker takes about  $Kb$  function evaluations per round.
- Fed-ZO-SGD. It is a zeroth-order extension of the standard federated SGD algorithm, where each worker's iteration uses an individually random minibatch of size  $b$ . Each worker's SGD runs for  $K' = K/2$  iterations with the step-size  $\alpha_k^t = \frac{\alpha_0^t}{\sqrt{k+1}} = \frac{\alpha}{\sqrt{(k+1)(t+1)}}$ . It uses the same setting for Gaussian smoothing as in Fed-ZO-GD.
- ZO-signSGD. This method is originally proposed in [20] and we adopt its variant with majority vote for distributed optimization. In each round, each worker computes  $K' = K/2$  gradient estimators, takes the sign of their average, and then uploads the result to the server. Each gradient estimator is obtained from a central-difference Gaussian smoothing over a minibatch batch of size  $b$ . The server performs global updates using the sign vector with step-size  $\alpha^t = \frac{\alpha}{\sqrt{t+1}}$ .
- ES-CSA. We use the standard  $(\mu; \lambda)$ -ES described in [27] with slight modifications for date decentralization. In each round, the server generates a population of  $\lambda$  solutions with a standard multivariate Gaussian distribution and broadcasts the whole population to each worker. The workers then evaluate the population with their local data. The server sums up, for each solution, the results collected from the workers and obtain the corresponding objective value. The best  $\lambda$  ones in the population are chosen and their recombination becomes the new population mean. In this setting, each worker takes  $\lambda \frac{N}{M}$  function evaluations per round. The standard cumulative step-size adaptation is used and the initial step-size is set to  $\alpha$ .

For the three gradient-based methods, we use the central-difference Gaussian smoothing which takes two function evaluations on each data sample; so the setting  $K' = K/2$  ensures that the total number of function evaluations per round and per worker is  $Kb$ , being consistent with DES. For CSA-ES, the population size is set to  $\lambda = MKb/N$ ; under this setting, all algorithms have exactly the same number of function evaluations per round.

For all algorithms, we choose  $b = 1000$ ,  $M = 10$ . We choose  $K = 100$  if  $n \leq 100$  and  $500$  if  $n > 100$ . Each algorithm is assigned with a budget of  $EN$  function evaluations, where  $E = 1000$  if  $n \leq 100$  and  $5000$  if  $n > 100$ . For algorithms relying on Gaussian smoothing, the finite-difference radius is  $\mu = 10^{-6}$ . The momentum parameter in DES is set to  $\beta = 0.5$ . All algorithms are run for 8 times individually for each pair of dataset and problem and the median results are reported. DES with the mixture Gaussian sampling and the mixture Rademacher sampling schemes

are denoted by DES-mG and DES-mR, respectively; their mixture parameter is set to  $l = 8$ .

## 5.2 Overall performance

We first test DES as well as the competitors on all three problems and over all six datasets. The initial step-size  $\alpha$  for each algorithm is chosen from  $\{0.1, 1, 10\}$  using a grid-search. Figures 1 and 2 report the convergence behavior of the algorithms, measured with the median training error versus the number of rounds. It is found that the DES methods with either standard Gaussian sampling or mixture sampling are the best performers in all cases. Belonging to the same ES family, our implementation of DES is significantly better than the non-distributed implementation of ES-CSA, where the latter performs the worst in most cases. This is caused by that the standard ES is for deterministic optimization and does not explore the stochastic characteristics of the objective function. Fed-ZO-SGD is the best one among the competitors and is competitive with DES in certain cases. Fed-ZO-GD, in most cases, is not competitive with DES, ZO-signSGD, or Fed-ZO-SGD.

We also observe that, when implemented in DES, the standard Gaussian sampling, the mixture Gaussian sampling, and the mixture Rademacher sampling do not show significant difference in performance. Although our analyses in Theorems 4 and 5 suggest the possibility that mixture sampling might degrade the convergence, the results here show that the degradation, if exists, is in general negligible. In many cases, in fact, mixture sampling can even improve the performance. This suggests that mixture sampling could be used as the default scheme for DES, given its availability in improving the sampling efficiency.

The experimental results obtained on testing sets are reported in the supplement. In general, the generalization performance of the DES methods are consistent with their training performance.

## 5.3 Adaptation of step-size

The theoretical analyses have demonstrated that DES converges with any initial step-size; and in this subsection we provide more empirical evidence. We first verify the performance of DES and the other competitors under different initial settings. In order to evaluate their performance over all problems and all datasets, we adopt the performance profile [54], a classic tool for visual comparison. The profile of an algorithm is the curve of the fraction of its solved test instances<sup>2</sup> (denoted by  $\rho(\tau)$ ) versus the amount of allocated computational budget (denoted by  $\tau$ ). The computational budget is measured by the ratio of the required number of rounds to that required by the best performer. We say an algorithm can solve a test instance if its obtained objective function value  $f'$  satisfies  $f(\mathbf{x}_0) - f' > \delta(f(\mathbf{x}_0) - f'_*)$  where  $\delta \in (0, 1)$  controls the accuracy and  $f'_*$  is the best objective value obtained among all algorithms. An algorithm with high values of  $\rho(\tau)$  or one that is located at the top left of the figure is preferable. In this section, the objective function value is measured by the training loss.

2. Test instance denotes the pair of problem and dataset.

Figure 3 plots the performance profiles of DES (with the standard Gaussian sampling) as well as the three competitors, with initial step-sizes chosen from  $\{0.1, 1, 10\}$ . We choose  $\delta = 0.1$  in plotting the profiles. The curves of DES are mostly lie to the left of the others, demonstrating that the relative performance of DES is in general robust to the step-size setting. For small  $\tau$ , the profile of DES with  $\alpha = 0.1$  lies to the right of Fed-ZO-SGD with  $\alpha = 10$ , and overlaps with that of the other methods; this indicates that  $\alpha = 0.1$  is too small for DES to achieve fast decrease in early stage. But when a sufficient amount of computation budget (e.g.,  $\tau \geq 10$ ) is allowed, then such a step-size setting can nevertheless lead to the performance comparable to Fed-ZO-SGD with the best tuned step-size. Fed-ZO-GD is not robust to the step-size setting. Its profile for  $\alpha = 0.1$  is not shown in the plot, implying that with this setting Fed-ZO-GD cannot solve any test instance.

Figure 4 provides, as an representative, the convergence trajectories of the algorithms with different initial step-sizes. In general, on the two convex problems (i.e., LR and LSVM), the performance of DES is quite insensitive to the initial step-size; all three settings admit approaching similar results in the long run. ES-CSA exhibits similar adaptation ability, albeit with relatively poor performance. The other gradient based methods are sensitive to step-size settings, leading to quite different solutions even in the convex problems. On the nonconvex problem NSVM, the initial value of the step-size seems to have a considerable influence on all methods, possibly because of that the step-size setting is critical in escaping local optima. In this case, large initial step-sizes seem to yield faster convergence, but may also lead to early stagnation.

## 5.4 Impact of momentum

The convergence rate established previously does not reflects its dependence on the momentum parameter, so here we investigate this empirically. Consider the mixture Rademacher sampling based DES method, with  $\beta$  chosen from  $\{0, 0.2, 0.4, 0.6, 0.8\}$  and  $\alpha$  fixed to 1. All other settings are the same as those in Section 5.2. Note that in the theoretical analyses we have required

$$\beta \leq \sqrt{\frac{1}{2\sqrt{2}}} \approx 0.6 \quad (15)$$

for technical reasons. So the choice  $\beta = 0.8$  is to verify whether the above requirement is necessary in practice.

Figure 5 gives the profile plot obtained on all test instances, measured with two different  $\delta$  settings. Note that the smaller  $\delta$  is, the higher solution-accuracy the curve reflects. It is found that the momentum mechanism becomes useless in the low accuracy domain; as setting  $\beta$  to 0 is enough to solve nearly 80% test instances within a very limited amount of computational budget. In this case, setting  $\beta$  to 0.8 is indeed harmful to the performance. To approach good performance in high accuracy, on the contrary, an appropriate setting of this parameter is generally helpful and could influence the final results. Again, we observe that the setting  $\beta = 0.8$  leads to poor performance, indicating that the assumption (15) seems to be mandatory. But it is worthy nothing that the choice of  $\beta$  is not critical to

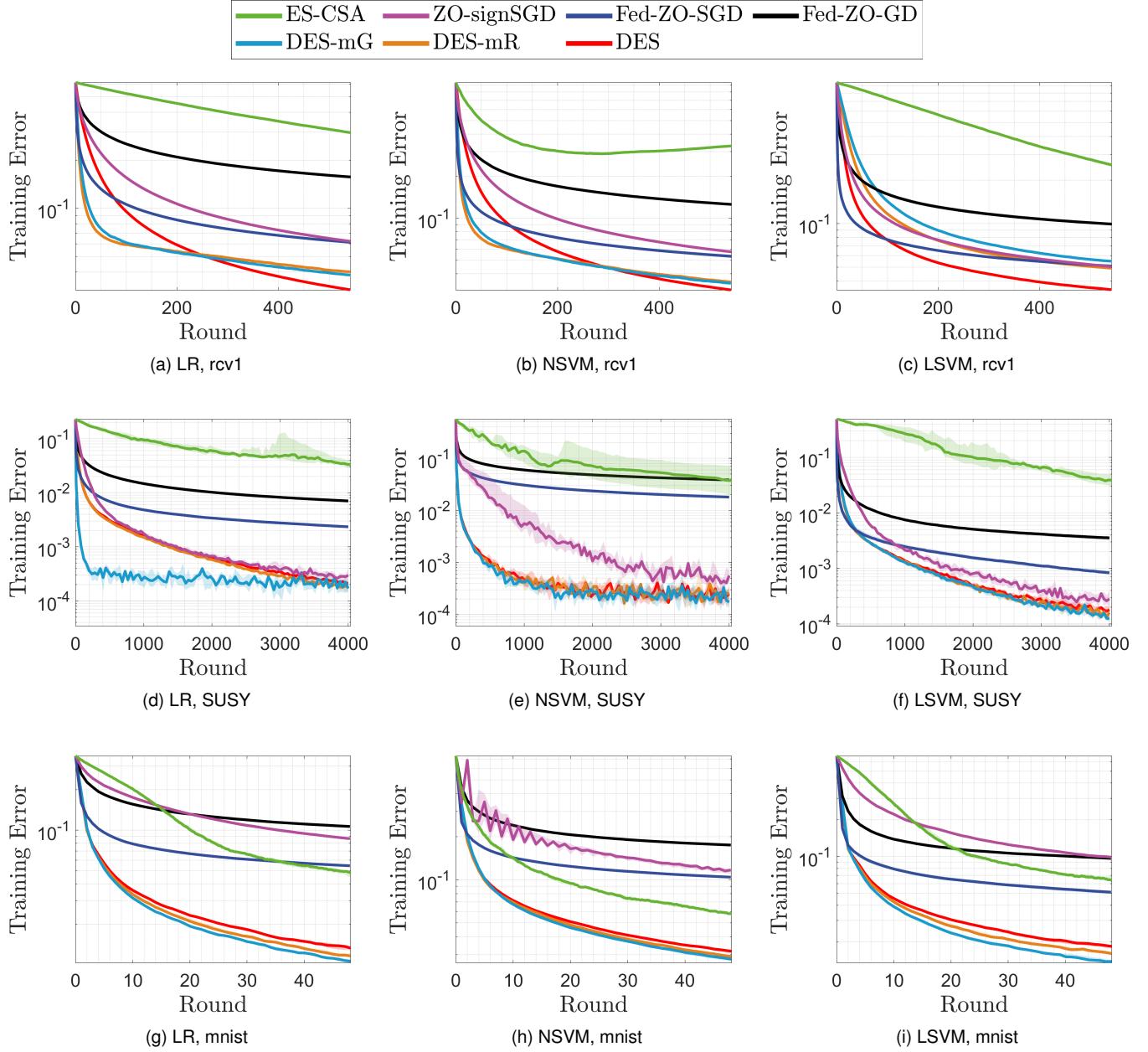


Fig. 1. Comparison on rcv1, SUSY, and mnist datasets. The curve displays the training error versus the number of rounds and the corresponding shaded area extends from the 25th to 75th percentiles over the results obtained from all independent runs.

the relative performance of DES compared with the other competitors; we suggest to fix its value in the range  $[0.2, 0.6]$  in all situations.

### 5.5 Impact of minibatch size

Here we verify the impact of minibatch size on the algorithm performance. We consider the mixture Rademacher sampling based DES method and choose  $\beta$  from  $\{100, 500, 1000, 1500, 2000\}$ . All other settings are the same as in Section 5.2.

Figure 6 reports the results obtained on all test instances via performance profile. It is clearly that whether minibatch size matters depends on which accuracy we would like to achieve. In the low accuracy case ( $\delta = 0.05$ ), choosing a small minibatch  $b = 100$  can solve at least 50% test instances very quickly, although suffering early termination later. In

this case, using a large minibatch does not lead to significant improvement in performance. Oppositely, the impact of minibatch size becomes quite clear in high accuracy case ( $\delta = 0.001$ ) where increasing  $b$  consistently improves the number of test instances that can be solved. This observation matches our theoretical analyses and suggests that a large minibatch is generally better if the computational cost is affordable at the worker-side.

## 6 CONCLUSION

In this work we propose the DES method via modifying the classic evolution strategy method and adapting it to the distributed setting. Our method uses a Gaussian probability model to guide the worker's local update, so it avoids finite-difference based smoothing techniques which might

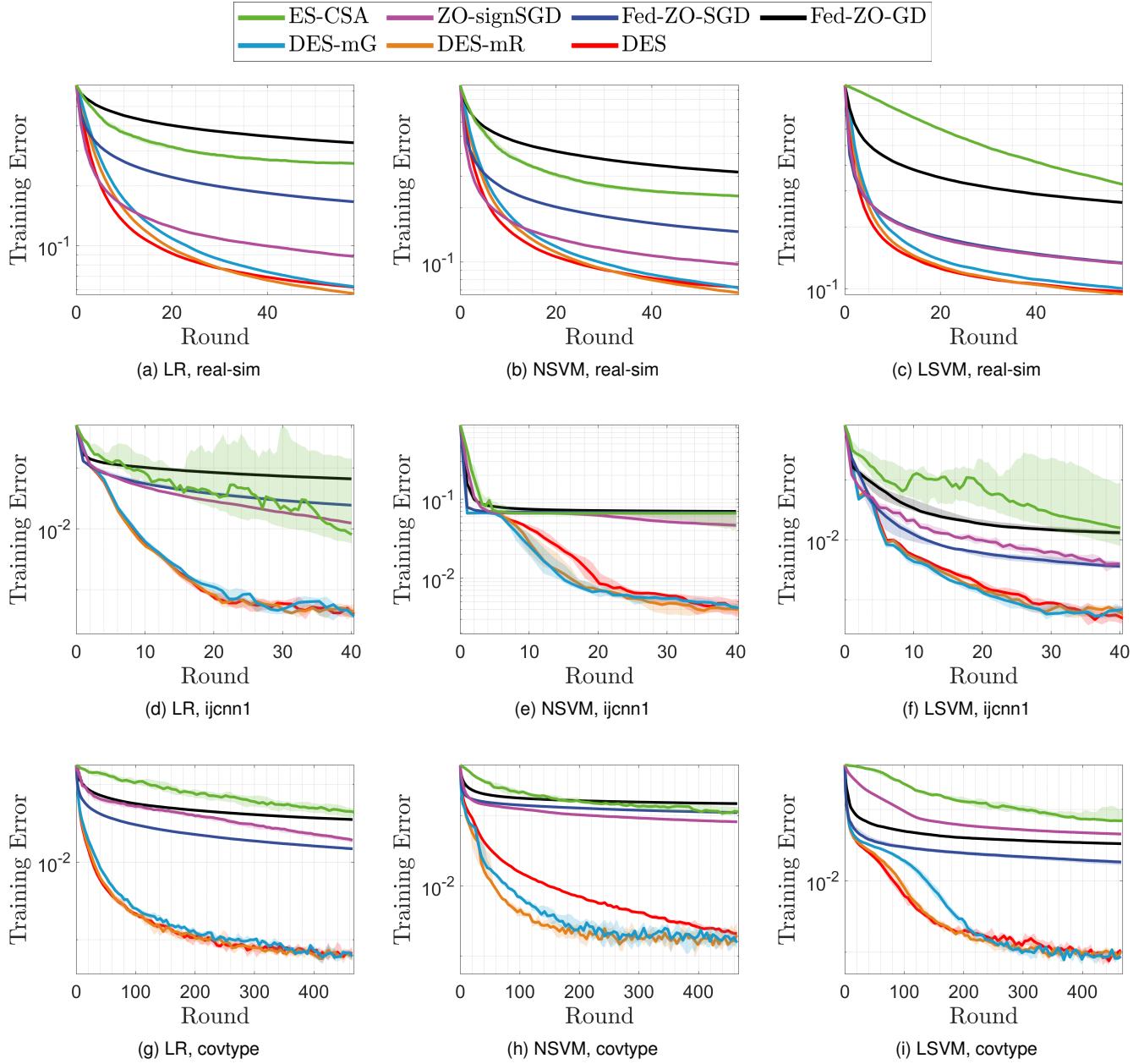


Fig. 2. Comparison on ijcnn, covtype, and real-sim datasets. The curve displays the training error versus the number of rounds and the corresponding shaded area extends from the 25th to 75th percentiles over the results obtained from all independent runs.

cause numerical issues. We have analyzed its convergence properties compared to existing zeroth-order and first-order methods, demonstrating its adaptivity to objective landscapes and the exploitation ability towards sparsity. Two alternative sampling schemes have been suggested and we find they lead to an improvement in sampling efficiency with no obvious degradation in performance. The current implementation of DES, however, does not support heterogeneous data distribution, which seems to be a common issue for those based on biased descent step; see [20], [25] for an example. The idea of bias correction suggested in [55] seems to address this issue, and is worth a try in further development of DES. This idea, nevertheless, would be incompatible with the comparison-based nature of the ES family. We would like to continue resolving this in the future. We hope our work on DES will serve as a starting

point for generalizing the rich studies in evolutionary computation communities to the distributed world.

## REFERENCES

- [1] H. Robbins and S. Monro, "A Stochastic Approximation Method," *The Annals of Mathematical Statistics*, vol. 22, no. 3, pp. 400–407, 1951.
- [2] L. Bottou, F. E. Curtis, and J. Nocedal, "Optimization Methods for Large-Scale Machine Learning," *SIAM Review*, vol. 60, no. 2, pp. 223–311, Jan. 2018.
- [3] S. Sun, Z. Cao, H. Zhu, and J. Zhao, "A Survey of Optimization Methods From a Machine Learning Perspective," *IEEE Transactions on Cybernetics*, vol. 50, no. 8, pp. 3668–3681, Aug. 2020.
- [4] M. Pereyra, P. Schniter, É. Chouzenoux, J.-C. Pesquet, J.-Y. Tourneret, A. O. Hero, III, and S. McLaughlin, "A Survey of Stochastic Simulation and Optimization Methods in Signal Processing," *IEEE Journal of Selected Topics in Signal Processing*, vol. 10, no. 2, pp. 224–241, 2016.

<span style="color:red">—</span> DES	<span style="color:black">—</span> Fed-ZO-GD	<span style="color:blue">—</span> Fed-ZO-SGD	<span style="color:magenta">—</span> ZO-signSGD
<span style="color:green">—</span> ES-CSA	<span style="color:purple">x</span> $\alpha = 1$	<span style="color:orange">□</span> $\alpha = 0.1$	<span style="color:darkblue">○</span> $\alpha = 10$

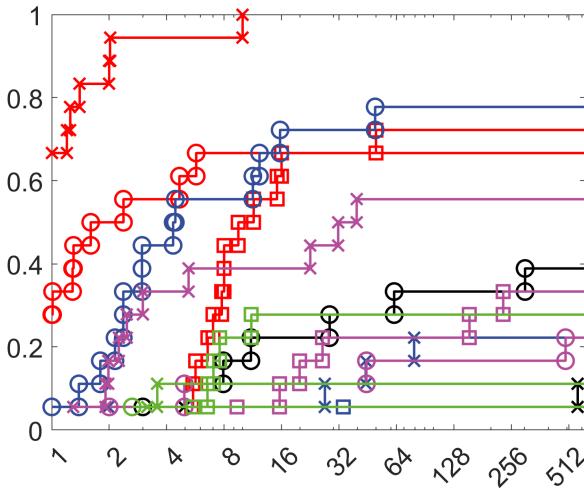


Fig. 3. Performance profiles ( $\delta = 0.1$ ) of different algorithms with different initial step-sizes. Results are obtained on all test instances.

- [5] D. Lee, N. He, P. Kamalaruban, and V. Cevher, "Optimization for Reinforcement Learning: From a single agent to cooperative agents," *IEEE Signal Processing Magazine*, vol. 37, no. 3, pp. 123–135, May 2020.
- [6] C. Gambella, B. Ghaddar, and J. Naoum-Sawaya, "Optimization problems for machine learning: A survey," *European Journal of Operational Research*, vol. 290, no. 3, pp. 807–828, May 2021.
- [7] F. E. Curtis and K. Scheinberg, "Adaptive Stochastic Optimization: A Framework for Analyzing Stochastic Optimization Algorithms," *IEEE Signal Processing Magazine*, vol. 37, no. 5, pp. 32–42, Sep. 2020.
- [8] A. Mokhtari and A. Ribeiro, "Stochastic Quasi-Newton Methods," *Proceedings of the IEEE*, vol. 108, no. 11, pp. 1906–1922, Nov. 2020.
- [9] J. Konečný, H. B. McMahan, F. X. Yu, P. Richtárik, A. T. Suresh, and D. Bacon, "Federated Learning: Strategies for Improving Communication Efficiency," *arXiv:1610.05492 [cs]*, Oct. 2016.
- [10] J. Konečný, H. B. McMahan, D. Ramage, and P. Richtárik, "Federated Optimization: Distributed Machine Learning for On-Device Intelligence," *arXiv:1610.02527 [cs]*, Oct. 2016.
- [11] X. Wang, Y. Han, C. Wang, Q. Zhao, X. Chen, and M. Chen, "In-Edge AI: Intelligentizing Mobile Edge Computing, Caching and Communication by Federated Learning," *IEEE Network*, vol. 33, no. 5, pp. 156–165, Sep. 2019.
- [12] A. G. Baydin, B. A. Pearlmutter, A. A. Radul, and J. M. Siskind, "Automatic Differentiation in Machine Learning: A Survey," *Journal of Machine Learning Research*, vol. 18, no. 153, pp. 1–43, 2018.
- [13] S. Liu, S. P. Chepuri, M. Fardad, E. Masazade, G. Leus, and P. K. Varshney, "Sensor Selection for Estimation with Correlated Measurement Noise," *IEEE Transactions on Signal Processing*, vol. 64, no. 13, pp. 3509–3522, Jul. 2016.
- [14] H. Kvamme, Ø. Borgan, and I. Scheel, "Time-to-Event Prediction with Neural Networks and Cox Regression." *J. Mach. Learn. Res.*, vol. 20, pp. 129:1–129:30, 2019.
- [15] Y. Nesterov and V. Spokoiny, "Random Gradient-Free Minimization of Convex Functions," *Foundations of Computational Mathematics*, vol. 17, no. 2, pp. 527–566, Apr. 2017.
- [16] J. Li, C. Wu, Z. Wu, and Q. Long, "Gradient-free method for non-smooth distributed optimization," *Journal of Global Optimization*, vol. 61, no. 2, pp. 325–340, Feb. 2015.
- [17] D. Yuan, S. Xu, and J. Lu, "Gradient-free method for distributed multi-agent optimization via push-sum algorithms," *International Journal of Robust and Nonlinear Control*, vol. 25, no. 10, pp. 1569–1580, 2015.
- [18] D. Yuan, D. W. C. Ho, and S. Xu, "Zeroth-Order Method for Distributed Optimization With Approximate Projections," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 27, no. 2, pp. 284–294, Feb. 2016.
- [19] B. Gu, Z. Huo, C. Deng, and H. Huang, "Faster Derivative-Free Stochastic Algorithm for Shared Memory Machines," in *International Conference on Machine Learning*. PMLR, Jul. 2018, pp. 1812–1821.
- [20] S. Liu, P.-Y. Chen, X. Chen, and M. Hong, "signSGD via Zeroth-Order Oracle," in *7th International Conference on Learning Representations*, New Orleans, LA, USA, May 2019.
- [21] A. K. Sahu and S. Kar, "Decentralized Zeroth-Order Constrained Stochastic Optimization Algorithms: Frank-Wolfe and Variants With Applications to Black-Box Adversarial Attacks," *Proceedings of the IEEE*, vol. 108, no. 11, pp. 1890–1905, Nov. 2020.
- [22] D. Wang, J. Yin, and W. Wang, "Distributed Randomized Gradient-Free Optimization Protocol of Multiagent Systems Over Weight-Unbalanced Digraphs." *IEEE Trans. Cybern.*, vol. 51, no. 1, pp. 473–482, 2021.
- [23] 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 Transactions on Information Theory*, vol. 61, no. 5, pp. 2788–2806, May 2015.
- [24] J. Nocedal and S. J. Wright, *Numerical Optimization*, 2nd ed., ser. Springer Series in Operations Research. New York: Springer, 2006.
- [25] J. Bernstein, Y.-X. Wang, K. Azizzadenesheli, and A. Anandkumar, "signSGD: Compressed Optimisation for Non-Convex Problems," in *International Conference on Machine Learning*, Jul. 2018, pp. 560–569.
- [26] H.-G. Beyer and H.-P. Schwefel, "Evolution strategies—A comprehensive introduction," *Natural computing*, vol. 1, no. 1, pp. 3–52, 2002.
- [27] N. Hansen, D. V. Arnold, and A. Auger, "Evolution strategies," in *Springer Handbook of Computational Intelligence*, J. Kacprzyk and W. Pedrycz, Eds. Springer Dordrecht Heidelberg London New York, 2015, pp. 871–898.
- [28] Z. Li, X. Lin, Q. Zhang, and H. Liu, "Evolution strategies for continuous optimization: A survey of the state-of-the-art," *Swarm and Evolutionary Computation*, vol. 56, p. 100694, Aug. 2020.
- [29] A. Auger and N. Hansen, "Linear Convergence of Comparison-based Step-size Adaptive Randomized Search via Stability of Markov Chains," *SIAM Journal on Optimization*, vol. 26, no. 3, pp. 1589–1624, Jan. 2016.
- [30] S. Astete-Morales, M.-L. Cauwet, and O. Teytaud, "Evolution Strategies with Additive Noise: A Convergence Rate Lower Bound," in *Proceedings of the 2015 ACM Conference on Foundations of Genetic Algorithms XIII*, ser. FOGA '15. New York, NY, USA: Association for Computing Machinery, Jan. 2015, pp. 76–84.
- [31] F. Zhou and G. Cong, "On the convergence properties of a k-step averaging stochastic gradient descent algorithm for nonconvex optimization," in *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI 2018, July 13–19, 2018, Stockholm, Sweden*, J. Lang, Ed. ijcai.org, 2018, pp. 3219–3227.
- [32] J. Zhang, C. De Sa, I. Mithagkas, and C. Ré, "Parallel SGD: When does averaging help?" *arXiv:1606.07365 [cs, stat]*, Jun. 2016.
- [33] J. Wang and G. Joshi, "Cooperative SGD: A unified Framework for the Design and Analysis of Communication-Efficient SGD Algorithms," *arXiv:1808.07576 [cs, stat]*, Jan. 2019.
- [34] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-Efficient Learning of Deep Networks from Decentralized Data," in *Artificial Intelligence and Statistics*, Apr. 2017, pp. 1273–1282.
- [35] T. Li, A. K. Sahu, A. Talwalkar, and V. Smith, "Federated Learning: Challenges, Methods, and Future Directions," *IEEE Signal Processing Magazine*, vol. 37, no. 3, pp. 50–60, May 2020.
- [36] X. Lian, Y. Huang, Y. Li, and J. Liu, "Asynchronous Parallel Stochastic Gradient for Nonconvex Optimization." in *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7–12, 2015, Montreal, Quebec, Canada*, 2015, pp. 2737–2745.
- [37] R. Ward, X. Wu, and L. Bottou, "AdaGrad Stepsizes: Sharp Convergence Over Nonconvex Landscapes," in *International Conference on Machine Learning*, May 2019, pp. 6677–6686.
- [38] S. J. Reddi, S. Kale, and S. Kumar, "On the Convergence of Adam and Beyond," in *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 – May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018.

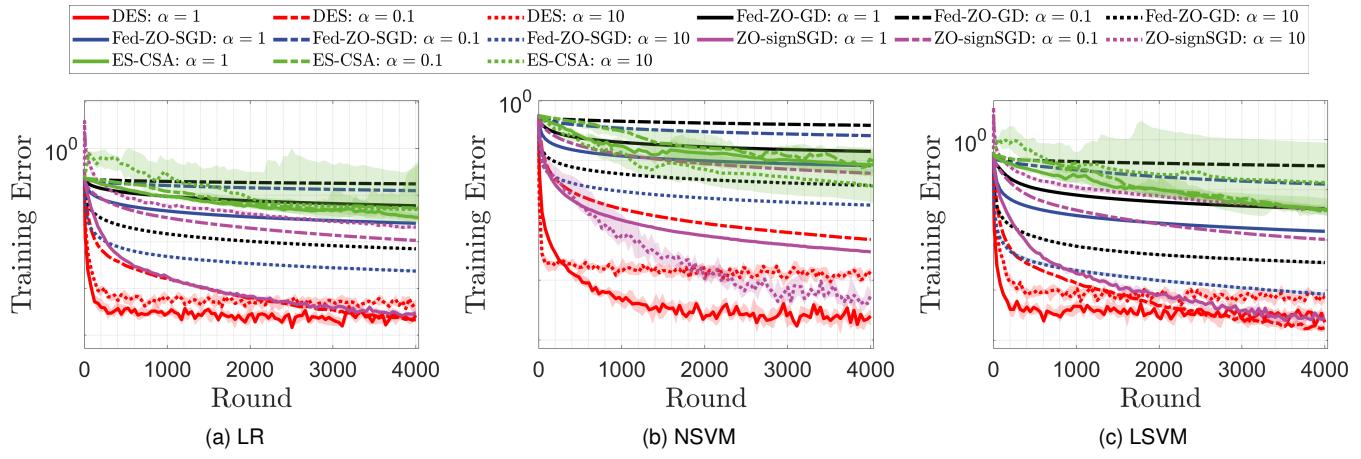
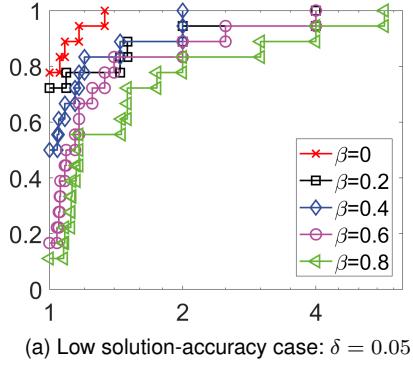
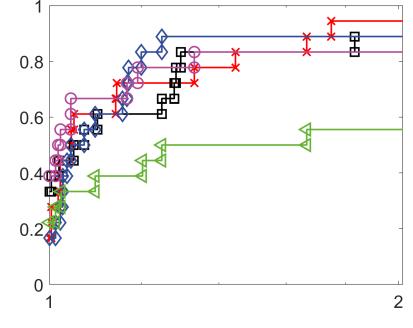


Fig. 4. Convergence on SUSY with different initial step-size settings. The curve displays the training error versus the number of rounds and the corresponding shaded area extends from the 25th to 75th percentiles over the results obtained from all independent runs.

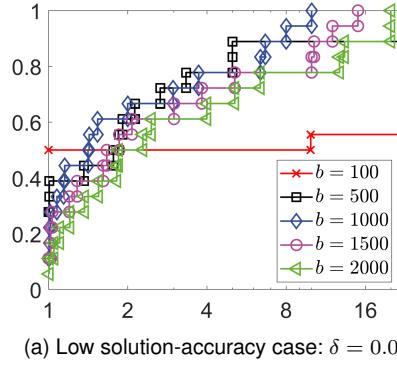


(a) Low solution-accuracy case:  $\delta = 0.05$

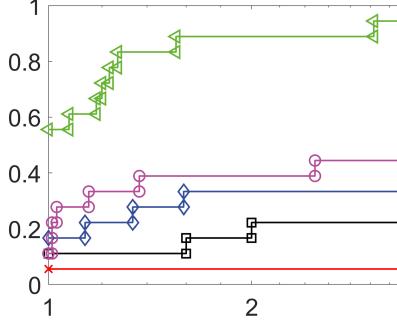


(b) High solution-accuracy case:  $\delta = 0.001$

Fig. 5. Performance profiles of DES with different momentum parameters. Results are obtained on all test instances. The mixture Rademacher sampling scheme is used in implementing DES.



(a) Low solution-accuracy case:  $\delta = 0.05$



(b) High solution-accuracy case:  $\delta = 0.001$

Fig. 6. Performance profiles of DES with different minibatch sizes. Results are obtained on all test instances. The mixture Rademacher sampling scheme is used in implementing DES.

- [39] K. Y. Levy, "Online to Offline Conversions, Universality and Adaptive Minibatch Sizes," in *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA, 2017*, pp. 1613–1622.
- [40] J. Duchi, E. Hazan, and Y. Singer, "Adaptive Subgradient Methods for Online Learning and Stochastic Optimization," *J. Mach. Learn. Res.*, vol. 12, pp. 2121–2159, Jul. 2011.
- [41] S. Reddi, Z. Charles, M. Zaheer, Z. Garrett, K. Rush, J. Konečný, S. Kumar, and H. B. McMahan, "Adaptive Federated Optimization," *arXiv:2003.00295 [cs, math, stat]*, Dec. 2020.
- [42] C. Xie, O. Koyejo, I. Gupta, and H. Lin, "Local AdaAlter: Communication-Efficient Stochastic Gradient Descent with Adaptive Learning Rates," in *12th Annual Workshop on Optimization for Machine Learning*, Dec. 2020.
- [43] Q. Tong, G. Liang, and J. Bi, "Effective Federated Adaptive Gradient Methods with Non-IID Decentralized Data," *arXiv:2009.06557 [cs, stat]*, Dec. 2020.
- [44] P. Rakshit, A. Konar, and S. Das, "Noisy evolutionary optimization algorithms – A comprehensive survey," *Swarm and Evolutionary Computation*, vol. 33, pp. 18–45, Apr. 2017.
- [45] H. Beyer and B. Sendhoff, "Toward a Steady-State Analysis of an Evolution Strategy on a Robust Optimization Problem With Noise-Induced Multimodality," *IEEE Transactions on Evolutionary Computation*, vol. 21, no. 4, pp. 629–643, Aug. 2017.
- [46] M. Hellwig and H.-G. Beyer, "On the steady state analysis of covariance matrix self-adaptation evolution strategies on the noisy ellipsoid model," *Theoretical Computer Science*, vol. 832, pp. 98–122, Sep. 2020.
- [47] C. Qian, Y. Yu, K. Tang, Y. Jin, X. Yao, and Z.-H. Zhou, "On the

- Effectiveness of Sampling for Evolutionary Optimization in Noisy Environments," *Evolutionary Computation*, vol. 26, no. 2, pp. 237–267, Jun. 2018.
- [48] Y.-J. Gong, W.-N. Chen, Z.-H. Zhan, J. Zhang, Y. Li, Q. Zhang, and J.-J. Li, "Distributed evolutionary algorithms and their models: A survey of the state-of-the-art," *Applied Soft Computing*, vol. 34, pp. 286–300, Sep. 2015.
- [49] T. Harada and E. Alba, "Parallel Genetic Algorithms: A Useful Survey," *ACM Computing Surveys*, vol. 53, no. 4, pp. 86:1–86:39, Aug. 2020.
- [50] Y. Akimoto, A. Auger, and T. Glasmachers, "Drift theory in continuous search spaces: Expected hitting time of the (1 + 1)-ES with 1/5 success rule," in *Proceedings of the Genetic and Evolutionary Computation Conference*. Kyoto Japan: ACM, Jul. 2018, pp. 801–808.
- [51] A. Agarwal, P. L. Bartlett, P. Ravikumar, and M. J. Wainwright, "Information-Theoretic Lower Bounds on the Oracle Complexity of Stochastic Convex Optimization," *IEEE Transactions on Information Theory*, vol. 58, no. 5, pp. 3235–3249, May 2012.
- [52] S. Ghadimi and G. Lan, "Stochastic First- and Zeroth-Order Methods for Nonconvex Stochastic Programming," *SIAM Journal on Optimization*, vol. 23, no. 4, pp. 2341–2368, Jan. 2013.
- [53] N. Hurley and S. Rickard, "Comparing Measures of Sparsity," *IEEE Transactions on Information Theory*, vol. 55, no. 10, pp. 4723–4741, 2009.
- [54] E. D. Dolan and J. J. Moré, "Benchmarking optimization software with performance profiles," *Mathematical Programming*, vol. 91, no. 2, pp. 201–213, Jan. 2002.
- [55] S. P. Karimireddy, Q. Rebjock, S. Stich, and M. Jaggi, "Error Feedback Fixes SignSGD and other Gradient Compression Schemes," in *International Conference on Machine Learning*. PMLR, May 2019, pp. 3252–3261.
- [56] E. Chlebus, "An approximate formula for a partial sum of the divergent p-series," *Applied Mathematics Letters*, vol. 22, no. 5, pp. 732–737, May 2009.

# Supplementary Appendices

## APPENDIX A

### PROOF OF THEOREM 1

*Proof.* For convenience define the following scalar operations

$$\text{sign}(a) = \begin{cases} 1 & \text{if } a \geq 0 \\ -1 & \text{if } a < 0 \end{cases} \quad \text{and} \quad \text{sign}_+(a) = \frac{\text{sign}(a) + 1}{2} = \begin{cases} 1 & \text{if } a \geq 0 \\ 0 & \text{if } a < 0 \end{cases}. \quad (16)$$

Note that the  $\text{sign}(\cdot)$  is different from the usual operation of taking sign, as in our definition it returns 1 when performed on 0. In addition, we have the following useful identities:

$$\text{sign}(a)b = (-1 + 2\mathbb{I}\{\text{sign}(a) = \text{sign}(b)\})|b| \quad (17)$$

and

$$\mathbb{I}\{\text{sign}(a) = \text{sign}(b)\} = \mathbb{I}\{|a + b| \geq |b|\} \quad (18)$$

which can be verified easily.

With the sign operation defined in (16), the iterations generated by Algorithm 1 can be rewritten as

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \text{sign}_+(f(\mathbf{x}_k) - f(\mathbf{x}_k + \alpha_k \mathbf{u}_k)) \mathbf{u}_k. \quad (19)$$

With Assumption 1, we can bound the per-iteration progress as

$$\begin{aligned} f(\mathbf{x}_{k+1}) - f(\mathbf{x}_k) &\leq \nabla f(\mathbf{x}_k)^T (\mathbf{x}_{k+1} - \mathbf{x}_k) + \frac{L}{2} \|\mathbf{x}_{k+1} - \mathbf{x}_k\|^2 \\ &\stackrel{(19)}{\leq} \alpha_k \text{sign}_+(f(\mathbf{x}_k) - f(\mathbf{x}_k + \alpha_k \mathbf{u}_k)) \nabla f(\mathbf{x}_k)^T \mathbf{u}_k + \frac{L\alpha_k^2}{2} \|\mathbf{u}_k\|^2 \\ &\stackrel{(16)}{=} \frac{1}{2} \alpha_k \mathbf{u}_k + \frac{1}{2} \alpha_k \underbrace{(\text{sign}(f(\mathbf{x}_k) - f(\mathbf{x}_k + \alpha_k \mathbf{u}_k))) \nabla f(\mathbf{x}_k)^T \mathbf{u}_k}_{\triangleq \mathfrak{A}} + \frac{L\alpha_k^2}{2} \|\mathbf{u}_k\|^2. \end{aligned}$$

Taking expectation with respect to  $\mathbf{u}_k$  at both sides, and according to Lemma 7, we have

$$\mathbb{E}_k [f(\mathbf{x}_{k+1})] - f(\mathbf{x}_k) \leq \frac{1}{2} \alpha_k \mathbb{E}_k [\mathfrak{A}] + \frac{L\alpha_k^2}{2} U \quad (20)$$

where  $\mathbb{E}_k$  denotes the expectation conditioned on the randomness at the  $k$ -th iteration.

We now bound the term  $\mathfrak{A}$  using identities (17) and (18):

$$\begin{aligned} \mathfrak{A} &\stackrel{(17)}{=} \left( -1 + 2\mathbb{I}\{\text{sign}(f(\mathbf{x}_k) - f(\mathbf{x}_k + \alpha_k \mathbf{u}_k)) = \text{sign}(\nabla f(\mathbf{x}_k)^T \mathbf{u}_k)\} \right) |\nabla f(\mathbf{x}_k)^T \mathbf{u}_k| \\ &= \left( -1 + 2\mathbb{I}\{\text{sign}(f(\mathbf{x}_k) - f(\mathbf{x}_k + \alpha_k \mathbf{u}_k)) = \text{sign}(\alpha_k \nabla f(\mathbf{x}_k)^T \mathbf{u}_k)\} \right) |\nabla f(\mathbf{x}_k)^T \mathbf{u}_k| \\ &\stackrel{(18)}{=} \left( -1 + 2\mathbb{I}\left\{ |f(\mathbf{x}_k + \alpha_k \mathbf{u}_k) - f(\mathbf{x}_k) - \alpha_k \nabla f(\mathbf{x}_k)^T \mathbf{u}_k| \geq \alpha_k |\nabla f(\mathbf{x}_k)^T \mathbf{u}_k| \right\} \right) |\nabla f(\mathbf{x}_k)^T \mathbf{u}_k| \\ &\leq \left( -1 + 2\mathbb{I}\left\{ \frac{L}{2} \|\alpha_k \mathbf{u}_k\|^2 \geq \alpha_k |\nabla f(\mathbf{x}_k)^T \mathbf{u}_k| \right\} \right) |\nabla f(\mathbf{x}_k)^T \mathbf{u}_k| \end{aligned} \quad (21)$$

where the last inequality is due to Assumption 1.

Substituting (21) into (20) gives

$$\begin{aligned} \mathbb{E}_k [f(\mathbf{x}_{k+1})] - f(\mathbf{x}_k) &\leq \frac{\alpha_k}{2} \mathbb{E}_k \left[ \left( -1 + 2\mathbb{I}\left\{ \frac{\alpha_k L}{2} \|\mathbf{u}_k\|^2 \geq |\nabla f(\mathbf{x}_k)^T \mathbf{u}_k| \right\} \right) |\nabla f(\mathbf{x}_k)^T \mathbf{u}_k| \right] + \frac{L\alpha_k^2}{2} U \\ &= -\frac{\alpha_k}{2} \mathbb{E}_k \left[ |\nabla f(\mathbf{x}_k)^T \mathbf{u}_k| \right] + \alpha_k \underbrace{\mathbb{E}_k \left[ \mathbb{I}\left\{ \frac{\alpha_k L}{2} \|\mathbf{u}_k\|^2 \geq |\nabla f(\mathbf{x}_k)^T \mathbf{u}_k| \right\} |\nabla f(\mathbf{x}_k)^T \mathbf{u}_k| \right]}_{\triangleq \mathfrak{B}} + \frac{L\alpha_k^2}{2} U \\ &= -\frac{\alpha_k}{\sqrt{2\pi}} \|\nabla f(\mathbf{x}_k)\|_2 + \alpha_k \mathfrak{B} + \frac{L\alpha_k^2}{2} U \end{aligned} \quad (22)$$

where the last equality uses the fact

$$\mathbb{E}[|\mathbf{y}^T \mathbf{u}|] = \sqrt{\frac{2}{\pi}} \|\mathbf{y}\|_2 \text{ for } \mathbf{u} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}). \quad (23)$$

Since the distribution of  $\mathbf{u}_k$  is isotropic, we can assume  $\nabla f(\mathbf{x}_k) = \|\nabla f(\mathbf{x}_k)\|_2 \mathbf{e}_1$  where  $\mathbf{e}_1 = (1, 0, \dots, 0)^T$ . Denoting  $u_{k,i}$  as the  $i$ -th element of  $\mathbf{u}_k$  and noting the assumption  $\|\cdot\| = \|\cdot\|_2$ , we have

$$\mathfrak{B} = \mathbb{E}_k \left[ \mathbb{I} \left\{ \frac{\alpha_k L}{2} \sum_{i=1}^n u_{k,i}^2 \geq \|\nabla f(\mathbf{x}_k)\|_2 |u_{k,1}| \right\} \|\nabla f(\mathbf{x}_k)\|_2 |u_{k,1}| \right]. \quad (24)$$

Now we decompose the expectation operation  $\mathbb{E}_k$  into two steps: firstly taking the expectation over  $u_{k,2}, \dots, u_{k,n}$  and secondly over  $u_{k,1}$ . That is,

$$\begin{aligned} \mathfrak{B} &= \mathbb{E}_{u_{k,1}} \mathbb{E}_{u_{k,2}, \dots, u_{k,n}} \left[ \mathbb{I} \left\{ \frac{\alpha_k L}{2} \sum_{i=1}^n u_{k,i}^2 \geq \|\nabla f(\mathbf{x}_k)\|_2 |u_{k,1}| \right\} \|\nabla f(\mathbf{x}_k)\|_2 |u_{k,1}| \right] \\ &= \mathbb{E}_{u_{k,1}} \left[ \mathbb{P}_{u_{k,2}, \dots, u_{k,n}} \left\{ \frac{\alpha_k L}{2} \sum_{i=1}^n u_{k,i}^2 \geq \|\nabla f(\mathbf{x}_k)\|_2 |u_{k,1}| \right\} \|\nabla f(\mathbf{x}_k)\|_2 |u_{k,1}| \right] \\ &\leq \mathbb{E}_{u_{k,1}} \left[ \frac{\alpha_k L}{2} \frac{u_{k,1}^2 + \sum_{i=2}^n \mathbb{E}_{u_{k,i}} [u_{k,i}^2]}{\|\nabla f(\mathbf{x}_k)\|_2 |u_{k,1}|} \|\nabla f(\mathbf{x}_k)\|_2 |u_{k,1}| \right] \\ &= \frac{\alpha_k L}{2} \mathbb{E}_{u_{k,1}} \left[ u_{k,1}^2 + \sum_{i=2}^n \mathbb{E}_{u_{k,i}} [u_{k,i}^2] \right] = \frac{\alpha_k L}{2} \mathbb{E}_k [\|\mathbf{u}_k\|^2]. \end{aligned}$$

Here we use the Markov inequality applied on the components  $u_{k,2}, \dots, u_{k,n}$ .

Substituting the above bound into (22) and using Lemma 7, we get

$$\mathbb{E}_k [f(\mathbf{x}_{k+1})] - f(\mathbf{x}_k) \leq -\frac{\alpha_k}{\sqrt{2\pi}} \|\nabla f(\mathbf{x}_k)\|_2 + L\alpha_k^2 U.$$

Taking the total expectation and summing over  $k = 0, 1, \dots, K-1$  give

$$\sum_{k=0}^{K-1} \alpha_k \mathbb{E} [\|\nabla f(\mathbf{x}_k)\|_2] \leq \sqrt{2\pi} \left( f(\mathbf{x}_0) - f_* + LU \sum_{k=0}^{K-1} \alpha_k^2 \right) \stackrel{(42)}{\leq} \sqrt{2\pi} (f(\mathbf{x}_0) - f_* + LU\alpha_0^2(1 + \log K)). \quad (25)$$

On the other hand, we can lower bound the left-hand side as

$$\sum_{k=0}^{K-1} \alpha_k \mathbb{E} [\|\nabla f(\mathbf{x}_k)\|_2] \stackrel{(44)}{\geq} \sqrt{K} \alpha_0 \left( \frac{1}{K} \sum_{k=0}^K \mathbb{E} [\|\nabla f(\mathbf{x}_k)\|_2] \right).$$

Combing this with (25) yields

$$\frac{1}{K} \sum_{k=0}^{K-1} \mathbb{E} [\|\nabla f(\mathbf{x}_k)\|_2] \leq \sqrt{\frac{2\pi}{K}} \left( \frac{f(\mathbf{x}_0) - f_*}{\alpha_0} + LU\alpha_0(1 + \log K) \right).$$

The bound (2) can be obtained via specifying  $U = n$  according to Lemma 7.  $\square$

## APPENDIX B

### A UNIFIED IMPLEMENTATION OF DES AND FUNDAMENTAL LEMMAS

Before proving the main results Theorems 2 to 5, we provide in this section some lemmas which will be used several times in the subsequent proofs. Since we have two DES implementations (i.e., Algorithms 2 and 3) and they only differ in the way of generating mutation vectors, we suggest to analyze them in a unified manner. To this end, we provide in Algorithm 4 a unified implementation of DES which can recover both Algorithm 2 and Algorithm 3. For example, it recovers Algorithm 2 if the mutation vector  $\mathbf{u}_{i,k}^t$  in Line 9 is drawn from the Gaussian distribution  $\mathcal{N}(\mathbf{0}, \mathbf{I})$ . It is also logically equivalent to Algorithm 3 when  $\mathbf{u}_{i,k}^t$  is drawn from the mixture Gaussian distribution  $\mathcal{M}_l^G$  or mixture Rademacher distribution  $\mathcal{M}_l^R$ . Note that the lemmas derived in this section do not rely on the detailed distribution for the mutation vectors. We will also not specify the vector norm when using the assumptions. The only requirement is that the variance of the mutation vector  $\mathbf{u}_{i,k}^t$  needs to be bounded by some constant  $U$  (see Line 9 in Algorithm 4). We will show in the next sections that this requirement indeed holds.

In the following we give some lemmas regarding the iterations generated from Algorithm 4. Due to the momentum mechanism, it is difficult to directly work with the solutions  $\{\mathbf{x}_t\}$ . Instead, we introduce a virtual sequence  $\{\mathbf{z}_t\}$  which can be regarded as a counterpart of  $\{\mathbf{x}_t\}$  without momentum:

$$\mathbf{z}_{t+1} = \frac{1}{1-\beta} \mathbf{x}_{t+1} - \frac{\beta}{1-\beta} \mathbf{x}_t.$$

**Algorithm 4** Unified implementation of DES for convergence analyses

---

**Require:**  $\mathbf{x}_0 \in \mathbb{R}^n$ : initial solution;  $\alpha \in \mathbb{R}_+$ : initial step-size;  $\beta \in [0, \sqrt{\frac{1}{2\sqrt{2}}}]$ : momentum parameter;  $b \geq \sqrt{T}$ : minibatch size;  
 $l \in \mathbb{Z}_+$ : mixture parameter

- 1: **for**  $t = 0, 1, \dots, T - 1$  **do**
- 2:   **for**  $i = 1, 2, \dots, M$  **in parallel do**
- 3:      $\mathbf{v}_{i,0}^t = \mathbf{x}_t$
- 4:      $\alpha_0^t = \alpha/(t+1)^{0.25}$
- 5:     Draw a minibatch  $\mathcal{D}_i$  of size  $b$
- 6:     Define  $f_i(\mathbf{x}) = \frac{1}{b} \sum_{\xi \in \mathcal{D}_i} F(\mathbf{x}; \xi)$
- 7:     **for**  $k = 0, 1, \dots, K - 1$  **do**
- 8:        $\alpha_k^t = \alpha_0^t/(k+1)^{0.5}$
- 9:       Generate a random vector  $\mathbf{u}_{i,k}^t$  satisfying  $\mathbb{E} [\|\mathbf{u}_{i,k}^t\|^2] \leq U$  for some positive constant  $U$  and some generic norm  $\|\cdot\|$
- 10:       $\mathbf{v}_{i,k+1}^t = \mathbf{v}_{i,k}^t + \alpha_k^t \text{sign}_+ (f_i(\mathbf{v}_{i,k}^t) - f_i(\mathbf{v}_{i,k}^t + \alpha_k^t \mathbf{u}_{i,k}^t))$  where  $\text{sign}_+$  is defined in (16)
- 11:     **end for**
- 12:   **end for**
- 13:    $\mathbf{d}_{t+1} = \frac{1}{M} \sum_{i=1}^M \mathbf{v}_{i,K}^t - \mathbf{x}_t$
- 14:    $\mathbf{m}_{t+1} = \beta \mathbf{m}_t + (1 - \beta) \mathbf{d}_{t+1}$
- 15:    $\mathbf{x}_{t+1} = \mathbf{x}_t + \mathbf{m}_{t+1}$
- 16: **end for**

---

To make it well-defined, we specify  $\mathbf{x}_{-1} = \mathbf{x}_0$  such that  $\mathbf{z}_0 = \mathbf{x}_0$ . We will characterize the algorithm behavior with  $\{\mathbf{z}_t\}$  and relate it to  $\{\mathbf{x}_t\}$  in the last step. Note that by this definition and according to the momentum rule (Lines 14-15 in Algorithm 4) we have

$$\mathbf{z}_{t+1} - \mathbf{z}_t = \mathbf{d}_{t+1} \quad \text{and} \quad \|\mathbf{x}_t - \mathbf{z}_t\| = \frac{\beta}{1 - \beta} \|\mathbf{x}_t - \mathbf{x}_{t-1}\|. \quad (26)$$

**Lemma 1.** *The descent step  $\mathbf{d}_{t+1}$  in Algorithm 4 can be bounded as*

$$\mathbb{E} [\|\mathbf{d}_{t+1}\|^2] \leq (\alpha_0^t)^2 UK (1 + \log K), \quad (27)$$

$$\mathbb{E} [\|\mathbf{d}_{t+1}\|] \leq 2\alpha_0^t \sqrt{KU}. \quad (28)$$

*Proof.* According to Line 13 of Algorithm 4 we have

$$\begin{aligned} \mathbb{E} [\|\mathbf{d}_{t+1}\|^2] &\stackrel{(*)}{\leq} \frac{1}{M} \sum_{i=1}^M \mathbb{E} [\|\mathbf{v}_{i,K}^t - \mathbf{x}_t\|^2] \\ &= \frac{1}{M} \sum_{i=1}^M \mathbb{E} \left[ \left\| \sum_{k=0}^{K-1} \mathbf{v}_{i,k+1}^t - \mathbf{v}_{i,k}^t \right\|^2 \right] \\ &\stackrel{(*)}{\leq} \frac{K}{M} \sum_{i=1}^M \sum_{k=0}^{K-1} \mathbb{E} [\|\mathbf{v}_{i,k+1}^t - \mathbf{v}_{i,k}^t\|^2] \\ &\leq \frac{K}{M} \sum_{i=1}^M \sum_{k=0}^{K-1} (\alpha_k^t)^2 \mathbb{E} [\|\mathbf{u}_{i,k}^t\|^2] \\ &\leq (\alpha_0^t)^2 \frac{K}{M} \sum_{i=1}^M \sum_{k=0}^{K-1} \frac{U}{k+1} \end{aligned}$$

where  $(*)$  is due to Jensen's inequality. Applying (42) in Lemma 8 gives (27).

Similarly, the bound (28) can be obtained as

$$\begin{aligned}
\mathbb{E} [\|\mathbf{d}_{t+1}\|] &\leq \frac{1}{M} \sum_{i=1}^M \mathbb{E} [\|\mathbf{v}_{i,K}^t - \mathbf{x}_t\|] \\
&= \frac{1}{M} \sum_{i=1}^M \mathbb{E} \left[ \left\| \sum_{k=0}^{K-1} \mathbf{v}_{i,k+1}^t - \mathbf{v}_{i,k}^t \right\| \right] \\
&\stackrel{(*)}{\leq} \frac{1}{M} \sum_{i=1}^M \sum_{k=0}^{K-1} \mathbb{E} [\|\mathbf{v}_{i,k+1}^t - \mathbf{v}_{i,k}^t\|] \\
&\leq \frac{1}{M} \sum_{i=1}^M \sum_{k=0}^{K-1} \alpha_k^t \mathbb{E} [\|\mathbf{u}_{i,k}^t\|] \\
&\leq \alpha_0^t \frac{1}{M} \sum_{i=1}^M \sum_{k=0}^{K-1} \sqrt{\frac{U}{k+1}}.
\end{aligned}$$

where  $(*)$  is due to Jensen's inequality and the last inequality is due to  $\mathbb{E} [\|\mathbf{u}_{i,k}^t\|] \leq \sqrt{\mathbb{E} [\|\mathbf{u}_{i,k}^t\|^2]} \leq \sqrt{U}$ . We can then reach (28) using (43) from Lemma 8.  $\square$

**Lemma 2.** Assume  $0 \leq \beta < \sqrt{\frac{1}{2\sqrt{2}}}$ . The change of the sequence  $\{\mathbf{x}_t\}$  in Algorithm 4 can be bounded as

$$\frac{1}{T} \sum_{t=0}^{T-1} \mathbb{E} [\|\mathbf{x}_t - \mathbf{x}_{t-1}\|] \leq \frac{160(1-\beta)\alpha\sqrt{KU}}{3T^{1/4}}, \quad (29)$$

$$\mathbb{E} [\|\mathbf{x}_t - \mathbf{x}_{t-1}\|^2] \leq \frac{(1-\beta)^2}{\frac{1}{2\sqrt{2}} - \beta^2} UK (1 + \log K) (\alpha_0^t)^2. \quad (30)$$

*Proof.* We first prove (29). By construction, we have for  $t > 1$

$$\|\mathbf{x}_t - \mathbf{x}_{t-1}\| = \|\mathbf{m}_t\| = \|\beta\mathbf{m}_{t-1} + (1-\beta)\mathbf{d}_t\| \leq \beta \|\mathbf{m}_{t-1}\| + (1-\beta) \|\mathbf{d}_t\|.$$

Expanding the above recursive bound gives

$$\|\mathbf{x}_t - \mathbf{x}_{t-1}\| \leq (\beta^{t-1} \|\mathbf{d}_1\| + \cdots + \beta \|\mathbf{d}_{t-1}\| + \|\mathbf{d}_t\|) (1-\beta).$$

Taking expectation at both sides yields

$$\begin{aligned}
\mathbb{E} [\|\mathbf{x}_t - \mathbf{x}_{t-1}\|] &\leq (1-\beta) \sum_{j=1}^t \beta^{t-j} \mathbb{E} [\|\mathbf{d}_j\|] \\
&\stackrel{(28)}{\leq} (1-\beta) \sum_{j=1}^t \beta^{t-j} 2\alpha_0^{j-1} \sqrt{KU} \\
&= 2(1-\beta)\alpha\sqrt{KU} \sum_{j=1}^t \frac{\beta^{t-j}}{j^{0.25}} \\
&\stackrel{(47)}{\leq} \frac{40(1-\beta)\alpha\sqrt{KU}}{t^{0.25}}
\end{aligned}$$

Recall that we have defined  $\mathbf{x}_0 = \mathbf{x}_{-1}$ , so

$$\frac{1}{T} \sum_{t=0}^{T-1} \mathbb{E} [\|\mathbf{x}_t - \mathbf{x}_{t-1}\|] \leq \frac{1}{T} \sum_{t=1}^{T-1} \frac{40(1-\beta)\alpha\sqrt{KU}}{t^{0.25}} \stackrel{(45)}{\leq} \frac{160(1-\beta)\alpha\sqrt{KU}}{3T^{1/4}}$$

and (29) is proved.

(30) is trivial for  $t = 0$ . For  $t \geq 1$ , it can be proved in a way similar to the above.

Firstly, we obtain via Jensen's inequality

$$\|\mathbf{x}_t - \mathbf{x}_{t-1}\|^2 = \|\mathbf{m}_t\|^2 = \|\beta\mathbf{m}_{t-1} + (1-\beta)\mathbf{d}_t\|^2 \leq 2\beta^2 \|\mathbf{m}_{t-1}\|^2 + 2(1-\beta)^2 \|\mathbf{d}_t\|^2.$$

Expanding the momentum terms  $\{\mathbf{m}_{t-1}\}$  and taking expectation give

$$\begin{aligned}
\mathbb{E} [\|\mathbf{x}_t - \mathbf{x}_{t-1}\|^2] &\leq 2(1-\beta)^2 \mathbb{E} [(2\beta^2)^{t-1} \|\mathbf{d}_1\|^2 + \cdots + (2\beta^2)^0 \|\mathbf{d}_t\|^2] \\
&= 2(1-\beta)^2 \sum_{j=1}^t (2\beta^2)^{t-j} \mathbb{E} [\|\mathbf{d}_j\|^2] \\
&\stackrel{(27)}{\leq} 2(1-\beta)^2 \sum_{j=1}^t (2\beta^2)^{t-j} (\alpha_0^{j-1})^2 UK (1 + \log K) \\
&= 2(1-\beta)^2 \sum_{j=1}^t \alpha^2 \frac{(2\beta^2)^{t-j}}{j^{0.5}} UK (1 + \log K) \\
&\stackrel{(48)}{\leq} \frac{2(1-\beta)^2 \alpha^2 UK (1 + \log K)}{\sqrt{t} (1 - 2\sqrt{2}\beta^2)} \\
&= \sqrt{\frac{t+1}{t}} \frac{2(1-\beta)^2 (\alpha_0^t)^2 UK (1 + \log K)}{1 - 2\sqrt{2}\beta^2}.
\end{aligned}$$

The last step is due to the definition of  $\alpha_0^t$ . Now use the assumption  $t \geq 1$  and we can reach (30).  $\square$

**Lemma 3.** Assume  $0 \leq \beta < \sqrt{\frac{1}{2\sqrt{2}}}$ . The worker drift in Algorithm 4 can be bounded as

$$\mathbb{E} [\|\mathbf{v}_{i,k}^t - \mathbf{z}_t\|^2] \leq \frac{2}{1 - 2\sqrt{2}\beta^2} UK (1 + \log K) (\alpha_0^t)^2. \quad (31)$$

*Proof.*

$$\begin{aligned}
\mathbb{E} [\|\mathbf{v}_{i,k}^t - \mathbf{z}_t\|^2] &\leq 2\mathbb{E} [\|\mathbf{v}_{i,k}^t - \mathbf{x}_t\|^2] + 2\mathbb{E} [\|\mathbf{x}_t - \mathbf{z}_t\|^2] \\
&\stackrel{(26)}{\leq} 2\mathbb{E} [\|\mathbf{v}_{i,k}^t - \mathbf{x}_t\|^2] + 2 \left( \frac{\beta}{1-\beta} \right)^2 \mathbb{E} [\|\mathbf{x}_t - \mathbf{x}_{t-1}\|^2] \\
&\stackrel{(30)}{\leq} 2\mathbb{E} [\|\mathbf{v}_{i,k}^t - \mathbf{x}_t\|^2] + \frac{2\beta^2}{\frac{1}{2\sqrt{2}} - \beta^2} UK (1 + \log K) (\alpha_0^t)^2
\end{aligned}$$

where

$$\begin{aligned}
\mathbb{E} [\|\mathbf{v}_{i,k}^t - \mathbf{x}_t\|^2] &\leq k \sum_{j=0}^{k-1} \mathbb{E} [\|\mathbf{v}_{i,j+1}^t - \mathbf{v}_{i,j}^t\|^2] \leq k \sum_{j=0}^{k-1} (\alpha_j^t)^2 \mathbb{E} [\|\mathbf{u}_{i,j}^t\|^2] \\
&\stackrel{(42)}{\leq} UK (\alpha_0^t)^2 (1 + \log k) \leq UK (\alpha_0^t)^2 (1 + \log K).
\end{aligned}$$

We thus obtain

$$\mathbb{E} [\|\mathbf{v}_{i,k}^t - \mathbf{z}_t\|^2] \leq \left( 2 + \frac{2\beta^2}{\frac{1}{2\sqrt{2}} - \beta^2} \right) UK (1 + \log K) (\alpha_0^t)^2 \leq \frac{2}{1 - 2\sqrt{2}\beta^2} UK (1 + \log K) (\alpha_0^t)^2. \quad \square$$

**Lemma 4.** Consider Algorithm 4. Let Assumptions 1 to 3 hold for some generic vector norm  $\|\cdot\|$ . Denote  $\mathbb{E}_{\mathcal{D}_i}$  as the expectation taken over the minibatch  $\mathcal{D}_i$ . We have

$$\begin{aligned}
\mathbb{E}_{\mathcal{D}_i} [\left| \nabla f(\mathbf{z}_t)^T \mathbf{u}_{i,k}^t \right| \mathbb{I} \left\{ \text{sign}(f_i(\mathbf{v}_{i,k}^t) - f_i(\mathbf{v}_{i,k}^t + \alpha_k^t \mathbf{u}_{i,k}^t)) = \text{sign}(\nabla f(\mathbf{z}_t)^T \mathbf{u}_{i,k}^t) \right\}] \\
\leq \frac{\alpha_k^t L + \omega_1 + \omega_2}{2} \|\mathbf{u}_{i,k}^t\|^2 + \frac{L^2}{2\omega_1} \|\mathbf{v}_{i,k}^t - \mathbf{z}_t\|^2 + \frac{\sigma^2}{2\omega_2 b}
\end{aligned} \quad (32)$$

*Proof.* Define

$$\mathfrak{A} = \left| \nabla f(\mathbf{z}_t)^T \mathbf{u}_{i,k}^t \right| \mathbb{I} \left\{ \text{sign}(f_i(\mathbf{v}_{i,k}^t) - f_i(\mathbf{v}_{i,k}^t + \alpha_k^t \mathbf{u}_{i,k}^t)) = \text{sign}(\nabla f(\mathbf{z}_t)^T \mathbf{u}_{i,k}^t) \right\}.$$

By (18), we have

$$\mathfrak{A} \stackrel{(18)}{=} \left| \nabla f(\mathbf{z}_t)^T \mathbf{u}_{i,k}^t \right| \mathbb{I} \left\{ \underbrace{\left| f_i(\mathbf{v}_{i,k}^t + \alpha_k^t \mathbf{u}_{i,k}^t) - f_i(\mathbf{v}_{i,k}^t) - \alpha_k^t \nabla f(\mathbf{z}_t)^T \mathbf{u}_{i,k}^t \right|}_{\triangleq \mathfrak{B}} \geq \alpha_k^t \left| \nabla f(\mathbf{z}_t)^T \mathbf{u}_{i,k}^t \right| \right\},$$

where

$$\begin{aligned} \mathfrak{B} &\leq \underbrace{\left| f_i(\mathbf{v}_{i,k}^t + \alpha_k^t \mathbf{u}_{i,k}^t) - f_i(\mathbf{v}_{i,k}^t) - \alpha_k^t \nabla f_i(\mathbf{v}_{i,k}^t)^T \mathbf{u}_{i,k}^t \right|}_{\mathfrak{C}_1} \\ &\quad + \underbrace{\alpha_k^t \left| \nabla f_i(\mathbf{v}_{i,k}^t)^T \mathbf{u}_{i,k}^t - \nabla f_i(\mathbf{z}_t)^T \mathbf{u}_{i,k}^t \right|}_{\mathfrak{C}_2} + \underbrace{\alpha_k^t \left| \nabla f_i(\mathbf{z}_t)^T \mathbf{u}_{i,k}^t - \nabla f(\mathbf{z}_t)^T \mathbf{u}_{i,k}^t \right|}_{\mathfrak{C}_3}. \end{aligned}$$

By Assumption 1 we have

$$\mathfrak{C}_1 \leq \frac{L}{2} \|\alpha_k^t \mathbf{u}_{i,k}^t\|^2.$$

Noting that we have  $|\mathbf{a}^T \mathbf{b}| \leq \frac{1}{2c} \|\mathbf{a}\|_*^2 + \frac{c}{2} \|\mathbf{b}\|^2$  for any  $\mathbf{a}, \mathbf{b} \in \mathbb{R}^n$  and  $c \in \mathbb{R}_+$ , so

$$\mathfrak{C}_2 \leq \frac{1}{2\omega_1} \|\nabla f_i(\mathbf{v}_{i,k}^t) - \nabla f_i(\mathbf{z}_t)\|_*^2 + \frac{\omega_1}{2} \|\mathbf{u}_{i,k}^t\|^2 \leq \frac{L^2}{2\omega_1} \|\mathbf{v}_{i,k}^t - \mathbf{z}_t\|^2 + \frac{\omega_1}{2} \|\mathbf{u}_{i,k}^t\|^2$$

and

$$\mathfrak{C}_3 \leq \frac{1}{2\omega_2} \|\nabla f_i(\mathbf{z}_t) - \nabla f(\mathbf{z}_t)\|_*^2 + \frac{\omega_2}{2} \|\mathbf{u}_{i,k}^t\|^2,$$

for some  $\omega_1, \omega_2 \in \mathbb{R}_+$ .

Putting all these together, we reach

$$\begin{aligned} \mathfrak{A} &= \left| \nabla f(\mathbf{z}_t)^T \mathbf{u}_{i,k}^t \right| \mathbb{I} \left\{ \mathfrak{B} \geq \alpha_k^t \left| \nabla f(\mathbf{z}_t)^T \mathbf{u}_{i,k}^t \right| \right\} \\ &\leq \left| \nabla f(\mathbf{z}_t)^T \mathbf{u}_{i,k}^t \right| \mathbb{I} \left\{ \frac{\alpha_k^t L + \omega_1 + \omega_2}{2} \|\mathbf{u}_{i,k}^t\|^2 + \frac{L^2}{2\omega_1} \|\mathbf{v}_{i,k}^t - \mathbf{z}_t\|^2 + \frac{\|\nabla f_i(\mathbf{z}_t) - \nabla f(\mathbf{z}_t)\|_*^2}{2\omega_2} \geq \left| \nabla f(\mathbf{z}_t)^T \mathbf{u}_{i,k}^t \right| \right\} \end{aligned}$$

Now take expectation over  $\mathcal{D}_i$ . Noting that Assumptions 2 and 3 indicate that the gradient variance can be scaled down by a factor of  $b = |\mathcal{D}_i|$ , so we have, based on the Markov inequality,

$$\begin{aligned} \mathbb{E}_{\mathcal{D}_i} [\mathfrak{A}] &\leq \left| \nabla f(\mathbf{z}_t)^T \mathbf{u}_{i,k}^t \right| \mathbb{P}_{\mathcal{D}_i} \left\{ \frac{\alpha_k^t L + \omega_1 + \omega_2}{2} \|\mathbf{u}_{i,k}^t\|^2 + \frac{L^2}{2\omega_1} \|\mathbf{v}_{i,k}^t - \mathbf{z}_t\|^2 + \frac{\|\nabla f_i(\mathbf{z}_t) - \nabla f(\mathbf{z}_t)\|_*^2}{2\omega_2} \geq \left| \nabla f(\mathbf{z}_t)^T \mathbf{u}_{i,k}^t \right| \right\} \\ &\leq \frac{\alpha_k^t L + \omega_1 + \omega_2}{2} \|\mathbf{u}_{i,k}^t\|^2 + \frac{L^2}{2\omega_1} \|\mathbf{v}_{i,k}^t - \mathbf{z}_t\|^2 + \frac{\mathbb{E}_{\mathcal{D}_i} [\|\nabla f_i(\mathbf{z}_t) - \nabla f(\mathbf{z}_t)\|_*^2]}{2\omega_2} \\ &\leq \frac{\alpha_k^t L + \omega_1 + \omega_2}{2} \|\mathbf{u}_{i,k}^t\|^2 + \frac{L^2}{2\omega_1} \|\mathbf{v}_{i,k}^t - \mathbf{z}_t\|^2 + \frac{\sigma^2}{2\omega_2 b}. \end{aligned}$$

□

## APPENDIX C PROOF OF THEOREMS 2 AND 3

In this section we proof the convergence results for Algorithm 2. Since Algorithm 2 is a special case of Algorithm 4 with Gaussian mutation, we can proceed in two steps. In the first step, we start from Lemmas 1, 3 and 4 (which are obtained for Algorithm 4) with the specification  $\mathbf{u}_{i,k}^t \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ . This admits bounding the gradient norm averaged over the virtual sequence  $\{\mathbf{z}_t\}$  with some constant  $U$ . The result is given in Lemma 5. Then, in the second step, we further specify the vector norm used in the assumptions, from which we can get the detailed values for  $U$ . In particular, based on Lemmas 2 and 5, we can prove Theorem 2 with the specification  $\|\cdot\| = \|\cdot\|_2$  and prove Theorem 3 with  $\|\cdot\| = \|\cdot\|_\infty$ .

**Lemma 5.** *Let Assumptions 1 to 3 hold for some generic vector norm  $\|\cdot\|$ . The virtual sequence  $\mathbf{z}_t$  produced by Algorithm 2 satisfies, for some  $U \geq \mathbb{E} [\|\mathbf{u}_{i,k}^t\|^2]$ ,*

$$\frac{1}{T} \sum_{t=0}^{T-1} \mathbb{E} [\|\nabla f(\mathbf{z}_t)\|_2] \leq \frac{\sqrt{2\pi}}{\alpha T^{3/4}} \left( \frac{f(\mathbf{x}_0) - f_*}{\sqrt{K}} + LU\Psi \sum_{t=0}^{T-1} (\alpha_0^t)^2 + 2\sigma \sqrt{\frac{U}{b} \sum_{t=0}^{T-1} \alpha_0^t} \right), \quad (33)$$

where  $\Psi$  is given in (7).

*Proof.* First rewrite  $\mathbb{E} \left[ \nabla f(\mathbf{z}_t)^T \mathbf{d}_{t+1} \right]$  as

$$\begin{aligned}
& \mathbb{E} \left[ \nabla f(\mathbf{z}_t)^T \mathbf{d}_{t+1} \right] \\
&= \mathbb{E} \left[ \nabla f(\mathbf{z}_t)^T \left( \frac{1}{M} \sum_{i=1}^M \mathbf{v}_{i,K}^t - \mathbf{x}_t \right) \right] \\
&= \frac{1}{M} \sum_{i=1}^M \sum_{k=0}^{K-1} \alpha_k^t \mathbb{E} \left[ \text{sign}_+ (f_i(\mathbf{v}_{i,k}^t) - f_i(\mathbf{v}_{i,k}^t + \alpha_k^t \mathbf{u}_{i,k}^t)) \nabla f(\mathbf{z}_t)^T \mathbf{u}_{i,k}^t \right] \\
&\stackrel{(16)}{=} \frac{1}{2M} \sum_{i=1}^M \sum_{k=0}^{K-1} \alpha_k^t \mathbb{E} \left[ (1 + \text{sign}(f_i(\mathbf{v}_{i,k}^t) - f_i(\mathbf{v}_{i,k}^t + \alpha_k^t \mathbf{u}_{i,k}^t))) \nabla f(\mathbf{z}_t)^T \mathbf{u}_{i,k}^t \right] \\
&\stackrel{(*)}{=} \frac{1}{2M} \sum_{i=1}^M \sum_{k=0}^{K-1} \alpha_k^t \mathbb{E} \left[ \text{sign}(f_i(\mathbf{v}_{i,k}^t) - f_i(\mathbf{v}_{i,k}^t + \alpha_k^t \mathbf{u}_{i,k}^t)) \nabla f(\mathbf{z}_t)^T \mathbf{u}_{i,k}^t \right] \\
&\stackrel{(17)}{=} \frac{1}{2M} \sum_{i=1}^M \sum_{k=0}^{K-1} \alpha_k^t \mathbb{E} \left[ \left| \nabla f(\mathbf{z}_t)^T \mathbf{u}_{i,k}^t \right| \left( -1 + 2\mathbb{I}\{\text{sign}(f_i(\mathbf{v}_{i,k}^t) - f_i(\mathbf{v}_{i,k}^t + \alpha_k^t \mathbf{u}_{i,k}^t)) = \text{sign}(\nabla f(\mathbf{z}_t)^T \mathbf{u}_{i,k}^t)\} \right) \right],
\end{aligned}$$

where  $(*)$  is due to  $\mathbb{E} [\mathbf{u}_{i,k}^t] = \mathbf{0}$ .

Now specify  $\mathbf{u}_{i,k}^t \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ . Using the identity in (23), we have

$$\begin{aligned}
& \mathbb{E} \left[ \nabla f(\mathbf{z}_t)^T \mathbf{d}_{t+1} \right] \\
&\stackrel{(23)}{\leq} -\frac{\mathbb{E} [\|\nabla f(\mathbf{z}_t)\|_2]}{\sqrt{2\pi}} \sum_{k=0}^{K-1} \alpha_k^t \\
&\quad + \frac{1}{M} \sum_{i=1}^M \sum_{k=0}^{K-1} \alpha_k^t \mathbb{E} \left[ \underbrace{\left| \nabla f(\mathbf{z}_t)^T \mathbf{u}_{i,k}^t \right| \mathbb{I}\{\text{sign}(f_i(\mathbf{v}_{i,k}^t) - f_i(\mathbf{v}_{i,k}^t + \alpha_k^t \mathbf{u}_{i,k}^t)) = \text{sign}(\nabla f(\mathbf{z}_t)^T \mathbf{u}_{i,k}^t)\}}_{\triangleq \mathfrak{A}} \right] \\
&\stackrel{(43)}{\leq} -\frac{\mathbb{E} [\|\nabla f(\mathbf{z}_t)\|_2]}{\sqrt{2\pi}} \alpha_0^t \sqrt{K} + \frac{1}{M} \sum_{i=1}^M \sum_{k=0}^{K-1} \alpha_k^t \mathbb{E} [\mathfrak{A}],
\end{aligned}$$

Now use Lemmas 4 and 7 to bound  $\mathbb{E} [\mathfrak{A}]$ :

$$\begin{aligned}
& \mathbb{E} \left[ \nabla f(\mathbf{z}_t)^T \mathbf{d}_{t+1} \right] + \frac{\mathbb{E} [\|\nabla f(\mathbf{z}_t)\|_2]}{\sqrt{2\pi}} \alpha_0^t \sqrt{K} \\
&\leq \frac{1}{2M} \sum_{i=1}^M \sum_{k=0}^{K-1} \alpha_k^t \left\{ (\alpha_k^t L + \omega_1 + \omega_2) U + \frac{L^2}{\omega_1} \mathbb{E} [\|\mathbf{v}_{i,k}^t - \mathbf{z}_t\|^2] + \frac{\sigma^2}{\omega_2 b} \right\} \\
&\leq \frac{L^2}{2M\omega_1} \sum_{i=1}^M \sum_{k=0}^{K-1} \alpha_k^t \mathbb{E} [\|\mathbf{v}_{i,k}^t - \mathbf{z}_t\|^2] + \frac{LU}{2} \sum_{k=0}^{K-1} (\alpha_k^t)^2 + \left( \frac{\omega_1 + \omega_2}{2} U + \frac{\sigma^2}{2\omega_2 b} \right) \sum_{k=0}^{K-1} \alpha_k^t \\
&\stackrel{(42,43)}{\leq} \frac{L^2}{2M\omega_1} \sum_{i=1}^M \sum_{k=0}^{K-1} \alpha_k^t \mathbb{E} [\|\mathbf{v}_{i,k}^t - \mathbf{z}_t\|^2] + \frac{LU}{2} (1 + \log K) (\alpha_0^t)^2 + \left( (\omega_1 + \omega_2) U + \frac{\sigma^2}{\omega_2 b} \right) \sqrt{K} \alpha_0^t
\end{aligned}$$

Using Lemma 3 and (43) yields

$$\begin{aligned}
& \mathbb{E} \left[ \nabla f(\mathbf{z}_t)^T \mathbf{d}_{t+1} \right] + \frac{\mathbb{E} [\|\nabla f(\mathbf{z}_t)\|_2]}{\sqrt{2\pi}} \alpha_0^t \sqrt{K} \\
&\leq LU \sqrt{K} \left( \frac{\alpha_0^t L}{\omega_1} \frac{2}{1 - 2\sqrt{2}\beta^2} K + \frac{1}{2\sqrt{K}} \right) (1 + \log K) (\alpha_0^t)^2 + \left( (\omega_1 + \omega_2) U + \frac{\sigma^2}{\omega_2 b} \right) \sqrt{K} \alpha_0^t
\end{aligned}$$

Consider now the setting  $\omega_1 = \frac{L\alpha_0^t}{\sqrt{K}}$ ,  $\omega_2 = \frac{\sigma}{\sqrt{Ub}}$ , and we can reach

$$\begin{aligned}
& \mathbb{E} \left[ \nabla f(\mathbf{z}_t)^T \mathbf{d}_{t+1} \right] \\
&\leq -\frac{\mathbb{E} [\|\nabla f(\mathbf{z}_t)\|_2]}{\sqrt{2\pi}} \alpha_0^t \sqrt{K} + LU \sqrt{K} \left( \left( \frac{2}{1 - 2\sqrt{2}\beta^2} \sqrt{K} + \frac{1}{2\sqrt{K}} \right) (1 + \log K) + \sqrt{K} \right) (\alpha_0^t)^2 + 2\sigma \sqrt{K} \sqrt{\frac{U}{b}} \alpha_0^t. \tag{34}
\end{aligned}$$

Using Assumption 1, we have

$$\begin{aligned} f(\mathbf{z}_{t+1}) &\leq f(\mathbf{z}_t) + \nabla f(\mathbf{z}_t)^T (\mathbf{z}_{t+1} - \mathbf{z}_t) + \frac{L}{2} \|\mathbf{z}_{t+1} - \mathbf{z}_t\|^2 \\ &\stackrel{(26)}{=} f(\mathbf{z}_t) + \nabla f(\mathbf{z}_t)^T \mathbf{d}_{t+1} + \frac{L}{2} \|\mathbf{d}_{t+1}\|^2 \end{aligned} \quad (35)$$

Taking total expectation, using (27) and (34), and rearranging yield

$$\begin{aligned} \frac{\mathbb{E}[\|\nabla f(\mathbf{z}_t)\|_2]}{\sqrt{2\pi}} \alpha_0^t &\leq \frac{\mathbb{E}[f(\mathbf{z}_t) - f(\mathbf{z}_{t+1})]}{\sqrt{K}} \\ &+ LU \left( \underbrace{\left( \left( \frac{2}{1-2\sqrt{2}\beta^2} + \frac{1}{2} \right) \sqrt{K} + \frac{1}{2\sqrt{K}} \right) (1 + \log K) + \sqrt{K}}_{\triangleq \Psi} \right) (\alpha_0^t)^2 + 2\sigma \sqrt{\frac{U}{b}} \alpha_0^t. \end{aligned}$$

Summing over  $t = 0, \dots, T-1$  gives

$$\sum_{t=0}^{T-1} \frac{\mathbb{E}[\|\nabla f(\mathbf{z}_t)\|_2]}{\sqrt{2\pi}} \alpha_0^t \leq \frac{f(\mathbf{z}_0) - f_*}{\sqrt{K}} + LU \Psi \sum_{t=0}^{T-1} (\alpha_0^t)^2 + 2\sigma \sqrt{\frac{U}{b}} \sum_{t=0}^{T-1} \alpha_0^t.$$

By (46), the left-hand side is no smaller than  $\frac{\alpha T^{3/4}}{\sqrt{2\pi}} \frac{1}{T} \sum_{t=0}^{T-1} \mathbb{E}[\|\nabla f(\mathbf{z}_t)\|_2]$ . And noting that, by definition,  $\mathbf{z}_0 = \mathbf{x}_0$ , we then obtain (33).  $\square$

**Proof of Theorem 2.** Under Assumption 1 and using the specification  $\|\cdot\| = \|\cdot\|_* = \|\cdot\|_2$ , we have

$$\|\nabla f(\mathbf{x}_t)\|_2 \leq \|\nabla f(\mathbf{x}_t) - \nabla f(\mathbf{z}_t)\|_2 + \|\nabla f(\mathbf{z}_t)\|_2 \leq L\|\mathbf{x}_t - \mathbf{z}_t\|_2 + \|\nabla f(\mathbf{z}_t)\|_2 \stackrel{(26)}{=} \frac{L\beta}{1-\beta} \|\mathbf{x}_t - \mathbf{x}_{t-1}\|_2 + \|\nabla f(\mathbf{z}_t)\|_2$$

which gives, via taking expectation,

$$\mathbb{E}[\|\nabla f(\mathbf{z}_t)\|_2] \geq \mathbb{E}[\|\nabla f(\mathbf{x}_t)\|_2] - \frac{L\beta}{1-\beta} \mathbb{E}[\|\mathbf{x}_t - \mathbf{x}_{t-1}\|_2].$$

Substituting this into (33) and using  $b \geq \sqrt{T}$  yield

$$\begin{aligned} \frac{1}{T} \sum_{t=0}^{T-1} \mathbb{E}[\|\nabla f(\mathbf{x}_t)\|_2] &\leq \frac{\sqrt{2\pi}}{\alpha T^{3/4}} \left( \frac{f(\mathbf{x}_0) - f_*}{\sqrt{K}} + LU \Psi \sum_{t=0}^{T-1} (\alpha_0^t)^2 + 2\sigma \sqrt{\frac{U}{b}} \sum_{t=0}^{T-1} \alpha_0^t \right) + \frac{L\beta}{1-\beta} \frac{1}{T} \sum_{t=0}^{T-1} \mathbb{E}[\|\mathbf{x}_t - \mathbf{x}_{t-1}\|_2] \\ &\stackrel{(29),(43),(45)}{\leq} \frac{\sqrt{2\pi}}{\alpha T^{3/4}} \left( \frac{f(\mathbf{x}_0) - f_*}{\sqrt{K}} + LU \Psi \alpha^2 2\sqrt{T} + 2\sigma \sqrt{\frac{U}{b}} \alpha \frac{4}{3} T^{3/4} \right) + L\beta \frac{160\alpha \sqrt{KU}}{3T^{1/4}} \\ &\leq \frac{\sqrt{2\pi}}{T^{3/4}} \frac{f(\mathbf{x}_0) - f_*}{\alpha \sqrt{K}} + \frac{\sqrt{U}}{T^{1/4}} \left( 2\alpha L \left( \sqrt{2\pi U} \Psi + \frac{80\beta \sqrt{K}}{3} \right) + \frac{8\sqrt{2\pi}\sigma}{3} \right). \end{aligned}$$

where when using (29) we have specified  $\|\cdot\| = \|\cdot\|_2$ . Finally, according to Lemma 7, we have  $\mathbb{E}[\|\mathbf{u}_{i,k}^t\|_2^2] = n$  when  $\mathbf{u}_{i,k}^t \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ . We can therefore choose  $U = n$  and then reach the target bound.  $\square$

**Proof of Theorem 3.** Firstly, the assumption  $\|\nabla f(\mathbf{x})\|_0 \leq s$  implies

$$\|\nabla f(\mathbf{x})\|_\infty \leq \|\nabla f(\mathbf{x})\|_2 \leq \|\nabla f(\mathbf{x})\|_1 \leq \sqrt{s} \|\nabla f(\mathbf{x})\|_\infty,$$

and hence we have

$$\begin{aligned} \|\nabla f(\mathbf{x}_t)\|_1 &\leq \|\nabla f(\mathbf{x}_t) - \nabla f(\mathbf{z}_t)\|_1 + \|\nabla f(\mathbf{z}_t)\|_1 \\ &\leq \|\nabla f(\mathbf{x}_t) - \nabla f(\mathbf{z}_t)\|_1 + \sqrt{s} \|\nabla f(\mathbf{z}_t)\|_2 \\ &\stackrel{(*)}{\leq} L \|\mathbf{x}_t - \mathbf{z}_t\|_\infty + \sqrt{s} \|\nabla f(\mathbf{z}_t)\|_2 \\ &\stackrel{(26)}{=} \frac{L\beta}{1-\beta} \|\mathbf{x}_t - \mathbf{x}_{t-1}\|_\infty + \sqrt{s} \|\nabla f(\mathbf{z}_t)\|_2 \end{aligned}$$

where (\*) uses Assumption 1 with the specification  $\|\cdot\| = \|\cdot\|_\infty$  and  $\|\cdot\|_* = \|\cdot\|_1$ . Taking expectation and rearranging give

$$\mathbb{E}[\|\nabla f(\mathbf{z}_t)\|_2] \geq \frac{1}{\sqrt{s}} \left( \mathbb{E}[\|\nabla f(\mathbf{x}_t)\|_1] - \frac{L\beta}{1-\beta} \mathbb{E}[\|\mathbf{x}_t - \mathbf{x}_{t-1}\|_\infty] \right).$$

Substituting this into the left-hand side of (33) in Lemma 5 yields

$$\begin{aligned}
\frac{1}{T} \sum_{t=0}^{T-1} \mathbb{E}[\|\nabla f(\mathbf{x}_t)\|_1] &\leq \frac{L\beta}{1-\beta} \frac{1}{T} \sum_{t=0}^{T-1} \mathbb{E}[\|\mathbf{x}_t - \mathbf{x}_{t-1}\|_\infty] + \frac{\sqrt{2\pi s}}{\alpha T^{3/4}} \left( \frac{f(\mathbf{x}_0) - f_*}{\sqrt{K}} + LU\Psi \sum_{t=0}^{T-1} (\alpha_0^t)^2 + 2\sigma \sqrt{\frac{U}{b}} \sum_{t=0}^{T-1} \alpha_0^t \right) \\
&\stackrel{(29)}{\leq} \frac{160L\beta\alpha\sqrt{KU}}{3T^{1/4}} + \frac{\sqrt{2\pi s}}{\alpha T^{3/4}} \left( \frac{f(\mathbf{x}_0) - f_*}{\sqrt{K}} + LU\Psi \sum_{t=0}^{T-1} (\alpha_0^t)^2 + 2\sigma \sqrt{\frac{U}{b}} \sum_{t=0}^{T-1} \alpha_0^t \right) \\
&\stackrel{(43,45)}{\leq} \frac{160L\beta\alpha\sqrt{KU}}{3T^{1/4}} + \frac{\sqrt{2\pi s}}{\alpha T^{3/4}} \left( \frac{f(\mathbf{x}_0) - f_*}{\sqrt{K}} + LU\Psi\alpha^2 2\sqrt{T} + 2\sigma \sqrt{\frac{U}{b}} \alpha \frac{4}{3} T^{3/4} \right) \\
&\leq \frac{\sqrt{2\pi s}}{T^{3/4}} \frac{f(\mathbf{x}_0) - f_*}{\alpha\sqrt{K}} + \frac{\sqrt{U}}{T^{1/4}} \left( 2\alpha L \left( \sqrt{2\pi s} \Psi + \frac{80\beta\sqrt{K}}{3} \right) + \frac{8\sqrt{2\pi s}\sigma}{3} \right)
\end{aligned}$$

where the last step uses the assumption  $b \geq \sqrt{T}$ .

Since we have used Lemma 5, we need  $U \geq \mathbb{E}[\|\mathbf{u}_{i,k}^t\|_\infty^2]$ . According to Lemma 7, we know  $U = 4\log(\sqrt{2}n)$  is valid choice. We then obtain the final bound as

$$\frac{1}{T} \sum_{t=0}^{T-1} \mathbb{E}[\|\nabla f(\mathbf{x}_t)\|_1] \leq \frac{\sqrt{2\pi s}}{T^{3/4}} \frac{f(\mathbf{x}_0) - f_*}{\alpha\sqrt{K}} + \frac{8\sqrt{\log(\sqrt{2}n)}}{T^{1/4}} \left( \alpha L \left( \sqrt{2\pi s} \log(\sqrt{2}n) \Psi + \frac{10\beta\sqrt{K}}{3} \right) + \frac{2\sqrt{2\pi s}\sigma}{3} \right)$$

□

## APPENDIX D

### PROOF OF PROPOSITIONS 1 AND 2

In the above proofs for DES with Gaussian mutation, we have repeatedly used the lower bound of  $\mathbb{E}[|\mathbf{u}^T \mathbf{y}|]$  where  $\mathbf{y} \in \mathbb{R}^n$  and  $\mathbf{u}$  is random. This bound is trivial when  $\mathbf{u} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ , as has been given in (23). To prove Theorems 4 and 5 we need a similar bound when  $\mathbf{u}$  is sampled from the mixture Gaussian distribution  $\mathcal{M}_l^G$  or the mixture Rademacher distribution  $\mathcal{M}_l^R$ . This can be achieved by analyzing the second-order and the fourth-order momentums of the corresponding probability distribution; this is the reason why Propositions 1 and 2 are required.

*Proof of Proposition 1.* We prove this proposition using moment-generating function.

Denote the moment-generating function of  $\mathcal{M}_l^G$  by  $M(\mathbf{t})$ . By definition,  $M(\mathbf{t})$  can be written as

$$\begin{aligned}
M(\mathbf{t}) &= \mathbb{E}[\exp(\mathbf{t}^T \mathbf{u})] = \mathbb{E}\left[\exp\sqrt{\frac{n}{l}} \left(\mathbf{t}^T \sum_{j=1}^l \mathbf{e}_{r_j} z_j\right)\right] = \mathbb{E}\left[\exp\sqrt{\frac{n}{l}} \left(\sum_{j=1}^l t_{r_j} z_j\right)\right] \stackrel{(*)}{=} \prod_{j=1}^l \mathbb{E}\left[\exp\left(\sqrt{\frac{n}{l}} t_{r_j} z_j\right)\right] \\
&= \prod_{j=1}^l \mathbb{E}\left[\sum_{k=1}^n \mathbb{I}\{r_j = k\} \exp\left(\sqrt{\frac{n}{l}} t_{r_j} z_j\right)\right] = \prod_{j=1}^l \sum_{k=1}^n \mathbb{P}\{r_j = k\} \mathbb{E}_k \left[\exp\left(\sqrt{\frac{n}{l}} t_k z_j\right)\right]
\end{aligned}$$

where  $t_{r_j}$  denotes the  $r_j$ -th element of  $\mathbf{t}$  and  $\mathbb{E}_k$  denotes the expectation conditioned on the event  $r_j = k$ . Equation (\*) is due to the independence of  $\{z_j\}$  and  $\{r_j\}$ .

Since the coordinate index  $r_j$  is sampled uniformly with replacement, we have  $\mathbb{P}\{r_j = k\} = \frac{1}{n}$ . Note that  $\mathbb{E}_k[\exp(t_k z_j)]$  is in fact the (conditioned) moment-generating function of the univariate Gaussian variable  $\sqrt{\frac{n}{l}} z_j$ , which is given by  $\exp\left(\frac{n}{2l} t_k^2\right)$ . So we reach

$$M(\mathbf{t}) = \prod_{j=1}^l \sum_{k=1}^n \frac{1}{n} \exp\left(\frac{n}{2l} t_k^2\right) = \left(\frac{1}{n} \sum_{k=1}^n \exp\left(\frac{n}{2l} t_k^2\right)\right)^l.$$

By construction, the covariance matrix must be diagonal, so we focus on its diagonal elements. Firstly, take the partial derivative with respect to  $t_j$  and this yields

$$\frac{\partial M(\mathbf{t})}{\partial t_j} = \frac{l}{n} \left( \frac{1}{n} \sum_{k=1}^n \exp\left(\frac{n}{2l} t_k^2\right) \right)^{l-1} \frac{\partial}{\partial t_j} \exp\left(\frac{n}{2l} t_j^2\right) = \left( \frac{1}{n} \sum_{k=1}^n \exp\left(\frac{n}{2l} t_k^2\right) \right)^{l-1} \exp\left(\frac{n}{2l} t_j^2\right) t_j.$$

The second-order partial derivative is then

$$\frac{\partial^2 M(\mathbf{t})}{\partial t_j^2} = \underbrace{\left\{ \frac{\partial}{\partial t_j} \left( \frac{1}{n} \sum_{k=1}^n \exp\left(\frac{n}{2l} t_k^2\right) \right)^{l-1} \right\}}_{T_1} \exp\left(\frac{n}{2l} t_j^2\right) t_j + \underbrace{\left( \frac{1}{n} \sum_{k=1}^n \exp\left(\frac{n}{2l} t_k^2\right) \right)^{l-1}}_{T_2} \underbrace{\frac{\partial}{\partial t_j} \left( \exp\left(\frac{n}{2l} t_j^2\right) t_j \right)}_{T_3}.$$

When setting  $\mathbf{t} = \mathbf{0}$ ,  $T_1$  vanishes and  $T_2$  becomes 1. We also have

$$T_3 = \exp\left(\frac{n}{2l}t_j^2\right) \left(\frac{\partial}{\partial t_j}\left(\frac{n}{2l}t_j^2\right)\right) t_j + \exp\left(\frac{n}{2l}t_j^2\right) \xrightarrow{\mathbf{t}=\mathbf{0}} 1.$$

So the  $j$ -th diagonal element is 1. We therefore conclude that  $\mathbf{u}$  has an identity covariance matrix.

In a similar manner, the moment-generating function of  $\mathbf{y}^T \mathbf{u}$  is

$$\tilde{M}(t) = \left(\frac{1}{n} \sum_{k=1}^n \exp\left(\frac{n}{2l}y_k^2 t^2\right)\right)^l.$$

Now expand the exponential term as Taylor series

$$\tilde{M}(t) = \left(\frac{1}{n} \sum_{k=1}^n \left(1 + \frac{n}{2l}y_k^2 t^2 + \frac{1}{2}\left(\frac{n}{2l}y_k^2 t^2\right)^2 + \mathcal{O}(t^6)\right)\right)^l = \left(1 + \frac{1}{2l}\|\mathbf{y}\|_2^2 t^2 + \frac{n}{8l^2}\|\mathbf{y}\|_4^4 t^4 + \mathcal{O}(t^6)\right)^l.$$

Using the multinomial theorem, we get

$$\begin{aligned} \tilde{M}(t) &= \sum_{j=0}^l \binom{l}{j} \left(\frac{1}{2l}\|\mathbf{y}\|_2^2 t^2 + \frac{n}{8l^2}\|\mathbf{y}\|_4^4 t^4 + \mathcal{O}(t^6)\right)^j \\ &= \binom{l}{1} \left(\frac{n}{8l^2}\|\mathbf{y}\|_4^4 t^4\right) + \binom{l}{2} \left(\frac{1}{2l}\|\mathbf{y}\|_2^2 t^2\right)^2 + 1 + At^2 + \mathcal{O}(t^6) \\ &= \frac{1}{8} \left(\frac{n}{l}\|\mathbf{y}\|_4^4 + \frac{l-1}{l}\|\mathbf{y}\|_2^4\right) t^4 + 1 + At^2 + \mathcal{O}(t^6) \end{aligned}$$

where  $A$  is some constant not depending on  $t$ . We can then reach the desired result by taking the fourth-order derivative and setting  $t = 0$ , i.e.,

$$\mathbb{E}[|\mathbf{y}^T \mathbf{u}|^4] = \frac{\partial^4}{\partial t^4} \tilde{M}(t) \Big|_{t=0} = 3 \left(\frac{n}{l}\|\mathbf{y}\|_4^4 + \frac{l-1}{l}\|\mathbf{y}\|_2^4\right) + \mathcal{O}(t^2) \Big|_{t=0} = 3 \left(\frac{n}{l}\|\mathbf{y}\|_4^4 + \frac{l-1}{l}\|\mathbf{y}\|_2^4\right).$$

□

**Proof of Proposition 2.** The proof is very similar to that of Proposition 1. First, we obtain the moment-generating function for  $\mathcal{M}_l^R$  as

$$\begin{aligned} M(\mathbf{t}) &= \mathbb{E}[\exp(\mathbf{t}^T \mathbf{u})] = \mathbb{E}\left[\exp\sqrt{\frac{n}{l}}\left(\mathbf{t}^T \sum_{j=1}^l \mathbf{e}_{r_j} z_j\right)\right] = \mathbb{E}\left[\exp\sqrt{\frac{n}{l}}\left(\sum_{j=1}^l t_{r_j} z_j\right)\right] \stackrel{(*)}{=} \prod_{j=1}^l \mathbb{E}\left[\exp\left(\sqrt{\frac{n}{l}}t_{r_j} z_j\right)\right] \\ &= \prod_{j=1}^l \mathbb{E}\left[\sum_{k=1}^n \mathbb{I}\{r_j = k\} \exp\left(\sqrt{\frac{n}{l}}t_{r_j} z_j\right)\right] = \prod_{j=1}^l \sum_{k=1}^n \mathbb{P}\{r_j = k\} \mathbb{E}_k\left[\exp\left(\sqrt{\frac{n}{l}}t_k z_j\right)\right] \\ &= \prod_{j=1}^l \frac{1}{2n} \sum_{k=1}^n \left(\exp\left(\sqrt{\frac{n}{l}}t_k\right) + \exp\left(-\sqrt{\frac{n}{l}}t_k\right)\right) = \left(\frac{1}{2n} \sum_{k=1}^n \left(\exp\left(\sqrt{\frac{n}{l}}t_k\right) + \exp\left(-\sqrt{\frac{n}{l}}t_k\right)\right)\right)^l. \end{aligned}$$

Equation (\*) in the above is due to the independence of  $\{z_j\}$  and  $\{r_j\}$ . The partial derivative with respect to  $t_j$  is then

$$\frac{\partial M(\mathbf{t})}{\partial t_j} = \frac{1}{2} \sqrt{\frac{l}{n}} \left(\frac{1}{2n} \sum_{k=1}^n \left(\exp\left(\sqrt{\frac{n}{l}}t_k\right) + \exp\left(-\sqrt{\frac{n}{l}}t_k\right)\right)\right)^{l-1} \left(\exp\left(\sqrt{\frac{n}{l}}t_j\right) - \exp\left(-\sqrt{\frac{n}{l}}t_j\right)\right)$$

and

$$\begin{aligned} \frac{\partial^2 M(\mathbf{t})}{\partial t_j^2} &= \frac{1}{2} \sqrt{\frac{l}{n}} \frac{\partial}{\partial t_j} \left(\frac{1}{2n} \sum_{k=1}^n \left(\exp\left(\sqrt{\frac{n}{l}}t_k\right) + \exp\left(-\sqrt{\frac{n}{l}}t_k\right)\right)\right)^{l-1} \underbrace{\left(\exp\left(\sqrt{\frac{n}{l}}t_j\right) - \exp\left(-\sqrt{\frac{n}{l}}t_j\right)\right)}_{=0 \text{ when } \mathbf{t}=\mathbf{0}} \\ &\quad + \frac{1}{2} \sqrt{\frac{l}{n}} \underbrace{\left(\frac{1}{2n} \sum_{k=1}^n \left(\exp\left(\sqrt{\frac{n}{l}}t_k\right) + \exp\left(-\sqrt{\frac{n}{l}}t_k\right)\right)\right)^{l-1}}_{=1 \text{ when } \mathbf{t}=\mathbf{0}} \frac{\partial}{\partial t_j} \left(\exp\left(\sqrt{\frac{n}{l}}t_j\right) - \exp\left(-\sqrt{\frac{n}{l}}t_j\right)\right). \end{aligned}$$

We therefore obtain

$$\frac{\partial^2 M(\mathbf{t})}{\partial t_j^2} \Big|_{\mathbf{t}=\mathbf{0}} = \frac{1}{2} \sqrt{\frac{l}{n}} \frac{\partial}{\partial t_j} \left(\exp\left(\sqrt{\frac{n}{l}}t_j\right) - \exp\left(-\sqrt{\frac{n}{l}}t_j\right)\right) \Big|_{\mathbf{t}=\mathbf{0}} = 1$$

As the covariance matrix is diagonal, we conclude from the above that the covariance matrix is an identity matrix.

The moment-generating function of the random variable  $\mathbf{y}^T \mathbf{u}$ , denoted by  $\tilde{M}(t)$ , can be obtained by substituting  $t = \mathbf{y}^T \mathbf{u}$  into  $M(t)$ :

$$\tilde{M}(t) = \left( \frac{1}{2n} \sum_{k=1}^n \left( \exp\left(\sqrt{\frac{n}{l}} y_k t\right) + \exp\left(-\sqrt{\frac{n}{l}} y_k t\right) \right) \right)^l = \left( \frac{1}{n} \sum_{k=1}^n \cosh\left(\sqrt{\frac{n}{l}} y_k t\right) \right)^l.$$

Now expanding the cosh function using Taylor series, we obtain

$$\begin{aligned} \tilde{M}(t) &= \left( \frac{1}{n} \sum_{k=1}^n \left( 1 + \frac{1}{2} \left( \sqrt{\frac{n}{l}} y_k t \right)^2 + \frac{1}{4!} \left( \sqrt{\frac{n}{l}} y_k t \right)^4 + \mathcal{O}(t^6) \right) \right)^l \\ &= \left( 1 + \frac{1}{2l} \|\mathbf{y}\|_2^2 t^2 + \frac{n}{24l^2} \|\mathbf{y}\|_4^4 t^4 + \mathcal{O}(t^6) \right)^l \\ &= l \left( \frac{n}{24l^2} \|\mathbf{y}\|_4^4 t^4 \right) + \frac{l(l-1)}{2} \left( \frac{1}{2l} \|\mathbf{y}\|_2^2 t^2 \right)^2 + 1 + At^2 + \mathcal{O}(t^6). \end{aligned}$$

The fourth-order moment of  $\mathbf{y}^T \mathbf{u}$  can be obtained as

$$\mathbb{E}[|\mathbf{y}^T \mathbf{u}|] = \frac{\partial^4}{\partial t^4} \tilde{M}(t) \Big|_{t=0} = \frac{n}{l} \|\mathbf{y}\|_4^4 + 3 \frac{l-1}{l} \|\mathbf{y}\|_2^4.$$

□

## APPENDIX E

### PROOF OF THEOREMS 4 AND 5

We will require the following lemma, which can be derived from Propositions 1 and 2.

**Lemma 6.** Let  $\mathbf{y} \in \mathbb{R}^n$  be a vector satisfying  $\|\mathbf{y}\|_2^4/\|\mathbf{y}\|_4^4 \geq \tilde{s}$  for some constant  $s \in [1, n]$ . We have

$$\mathbb{E}[|\mathbf{y}^T \mathbf{u}|] \geq \frac{\|\mathbf{y}\|_2}{\sqrt{3n/(\tilde{s}l) + 3}} \quad \text{for } \mathbf{u} \sim \mathcal{M}_l^G$$

and

$$\mathbb{E}[|\mathbf{y}^T \mathbf{u}|] \geq \frac{\|\mathbf{y}\|_2}{\sqrt{n/(\tilde{s}l) + 3}} \quad \text{for } \mathbf{u} \sim \mathcal{M}_l^R.$$

*Proof.* First, by Hölder's inequality, we have

$$\mathbb{E}[|\mathbf{y}^T \mathbf{u}|] \geq \frac{(\mathbb{E}[|\mathbf{y}^T \mathbf{u}|^2])^{3/2}}{(\mathbb{E}[|\mathbf{y}^T \mathbf{u}|^4])^{1/2}} = \frac{\|\mathbf{y}\|_2^3}{(\mathbb{E}[|\mathbf{y}^T \mathbf{u}|^4])^{1/2}}. \quad (36)$$

where the equality uses the fact  $\mathbb{V}[\mathbf{u}] = \mathbf{I}$ , according to Propositions 1 and 2.

Now consider the case of mixture Gaussian sampling. In this case, we have, from (11), that

$$\mathbb{E}[|\mathbf{y}^T \mathbf{u}|] \geq \frac{\|\mathbf{y}\|_2^3}{\sqrt{3 \left( \frac{n}{l} \|\mathbf{y}\|_4^4 + \frac{l-1}{l} \|\mathbf{y}\|_2^4 \right)}}. \quad (37)$$

Using the assumption  $\|\mathbf{y}\|_2^4/\|\mathbf{y}\|_4^4 \geq \tilde{s}$  then yields

$$\mathbb{E}[|\mathbf{y}^T \mathbf{u}|] \geq \frac{\|\mathbf{y}\|_2^3}{\sqrt{3 \left( \frac{n}{\tilde{s}l} \|\mathbf{y}\|_2^4 + \frac{l-1}{l} \|\mathbf{y}\|_2^4 \right)}} = \frac{\|\mathbf{y}\|_2}{\sqrt{3 \left( \frac{n}{\tilde{s}l} + \frac{l-1}{l} \right)}} \geq \frac{\|\mathbf{y}\|_2}{\sqrt{3(n/(\tilde{s}l) + 1)}}.$$

Consider then the case of mixture Rademacher sampling. From (36), (12), and the assumption  $\|\mathbf{y}\|_2^4/\|\mathbf{y}\|_4^4 \geq \tilde{s}$ , we have

$$\mathbb{E}[|\mathbf{y}^T \mathbf{u}|] \geq \frac{\|\mathbf{y}\|_2^3}{\sqrt{\frac{n}{l} \|\mathbf{y}\|_4^4 + 3 \frac{l-1}{l} \|\mathbf{y}\|_2^4}} \geq \frac{\|\mathbf{y}\|_2^3}{\sqrt{\frac{n}{\tilde{s}l} \|\mathbf{y}\|_2^4 + 3 \frac{l-1}{l} \|\mathbf{y}\|_2^4}} = \frac{\|\mathbf{y}\|_2}{\sqrt{n/(\tilde{s}l) + 3 \frac{l-1}{l}}} \geq \frac{\|\mathbf{y}\|_2}{\sqrt{n/(\tilde{s}l) + 3}}. \quad (38)$$

□

**Proof of Theorem 4.** Recall that the DES with mixture Gaussian sampling is a special case of Algorithm 4, so we can reuse Lemmas 1 to 4 which are derived for Algorithm 4.

The first step in this proof is to obtain a similar bound as in Lemma 5. We begin with rewriting  $\mathbb{E} \left[ \nabla f(\mathbf{x}_t)^T \mathbf{d}_{t+1} \right]$ . For  $\beta = 0$ , we have  $\mathbf{z}_t = \mathbf{x}_t$  and

$$\begin{aligned}
& \mathbb{E} \left[ \nabla f(\mathbf{x}_t)^T \mathbf{d}_{t+1} \right] \\
&= \mathbb{E} \left[ \nabla f(\mathbf{x}_t)^T \left( \frac{1}{M} \sum_{i=1}^M \mathbf{v}_{i,K}^t - \mathbf{x}_t \right) \right] \\
&= \frac{1}{M} \sum_{i=1}^M \sum_{k=0}^{K-1} \alpha_k^t \mathbb{E} \left[ \text{sign}_+ (f_i(\mathbf{v}_{i,k}^t) - f_i(\mathbf{v}_{i,k}^t + \alpha_k^t \mathbf{u}_{i,k}^t)) \nabla f(\mathbf{x}_t)^T \mathbf{u}_{i,k}^t \right] \\
&\stackrel{(16)}{=} \frac{1}{2M} \sum_{i=1}^M \sum_{k=0}^{K-1} \alpha_k^t \mathbb{E} \left[ (1 + \text{sign}(f_i(\mathbf{v}_{i,k}^t) - f_i(\mathbf{v}_{i,k}^t + \alpha_k^t \mathbf{u}_{i,k}^t))) \nabla f(\mathbf{x}_t)^T \mathbf{u}_{i,k}^t \right] \\
&= \frac{1}{2M} \sum_{i=1}^M \sum_{k=0}^{K-1} \alpha_k^t \mathbb{E} \left[ \text{sign}(f_i(\mathbf{v}_{i,k}^t) - f_i(\mathbf{v}_{i,k}^t + \alpha_k^t \mathbf{u}_{i,k}^t)) \nabla f(\mathbf{x}_t)^T \mathbf{u}_{i,k}^t \right] \\
&\stackrel{(17)}{=} \frac{1}{2M} \sum_{i=1}^M \sum_{k=0}^{K-1} \alpha_k^t \mathbb{E} \left[ \left| \nabla f(\mathbf{x}_t)^T \mathbf{u}_{i,k}^t \right| \left( -1 + 2\mathbb{I}\{\text{sign}(f_i(\mathbf{v}_{i,k}^t) - f_i(\mathbf{v}_{i,k}^t + \alpha_k^t \mathbf{u}_{i,k}^t)) = \text{sign}(\nabla f(\mathbf{x}_t)^T \mathbf{u}_{i,k}^t)\} \right) \right] \\
&\stackrel{(44)}{\leq} -\frac{\alpha_0^t}{2M\sqrt{K}} \sum_{i=1}^M \sum_{k=0}^{K-1} \mathbb{E} \left[ \left| \nabla f(\mathbf{x}_t)^T \mathbf{u}_{i,k}^t \right| \right] \\
&\quad + \frac{1}{M} \sum_{i=1}^M \sum_{k=0}^{K-1} \alpha_k^t \mathbb{E} \left[ \underbrace{\left| \nabla f(\mathbf{x}_t)^T \mathbf{u}_{i,k}^t \right| \mathbb{I}\{\text{sign}(f_i(\mathbf{v}_{i,k}^t) - f_i(\mathbf{v}_{i,k}^t + \alpha_k^t \mathbf{u}_{i,k}^t)) = \text{sign}(\nabla f(\mathbf{x}_t)^T \mathbf{u}_{i,k}^t)\}}_{\triangleq \mathfrak{A}} \right] \\
&= -\frac{\alpha_0^t}{2M\sqrt{K}} \sum_{i=1}^M \sum_{k=0}^{K-1} \mathbb{E} \left[ \left| \nabla f(\mathbf{x}_t)^T \mathbf{u}_{i,k}^t \right| \right] + \frac{1}{M} \sum_{i=1}^M \sum_{k=0}^{K-1} \alpha_k^t \mathbb{E} [\mathfrak{A}]
\end{aligned} \tag{39}$$

Using the assumption  $\|\nabla f(\mathbf{x})\|_2^4 / \|\nabla f(\mathbf{x})\|_4^4 \geq \tilde{s}$  and Lemma 6, we have

$$\mathbb{E} \left[ \nabla f(\mathbf{x}_t)^T \mathbf{d}_{t+1} \right] \leq -\frac{\alpha_0^t \sqrt{K}}{2V} \mathbb{E} [\|\nabla f(\mathbf{x}_t)\|_2] + \frac{1}{M} \sum_{i=1}^M \sum_{k=0}^{K-1} \alpha_k^t \mathbb{E} [\mathfrak{A}]$$

where  $V$  is a constant that can be set to

$$V = \sqrt{3 + 3n/(\tilde{s}l)}. \tag{40}$$

Note that Lemma 4 gives an upper bound for the term  $\mathbb{E}[\mathfrak{A}]$ . We therefore have

$$\begin{aligned}
& \mathbb{E} \left[ \nabla f(\mathbf{x}_t)^T \mathbf{d}_{t+1} \right] + \frac{\alpha_0^t \sqrt{K}}{2V} \mathbb{E} [\|\nabla f(\mathbf{x}_t)\|_2] \\
&\leq \frac{1}{M} \sum_{i=1}^M \sum_{k=0}^{K-1} \alpha_k^t \left( \frac{\alpha_k^t L + \omega_1 + \omega_2}{2} \mathbb{E} [\|\mathbf{u}_{i,k}^t\|^2] + \frac{L^2}{2\omega_1} \mathbb{E} [\|\mathbf{v}_{i,k}^t - \mathbf{z}_t\|^2] + \frac{\sigma^2}{2\omega_2 b} \right) \\
&\leq \frac{1}{M} \sum_{i=1}^M \sum_{k=0}^{K-1} \alpha_k^t \left( \frac{\alpha_k^t L + \omega_1 + \omega_2}{2} U + \frac{L^2}{2\omega_1} \mathbb{E} [\|\mathbf{v}_{i,k}^t - \mathbf{z}_t\|^2] + \frac{\sigma^2}{2\omega_2 b} \right).
\end{aligned}$$

Now use Lemma 3 to bound  $\mathbb{E} [\|\mathbf{v}_{i,k}^t - \mathbf{z}_t\|^2]$  and use the setting  $\beta = 0$ :

$$\begin{aligned}
& \mathbb{E} \left[ \nabla f(\mathbf{x}_t)^T \mathbf{d}_{t+1} \right] + \frac{\alpha_0^t \sqrt{K}}{2V} \mathbb{E} [\|\nabla f(\mathbf{x}_t)\|_2] \\
&\stackrel{(31), \beta=0}{\leq} \sum_{k=0}^{K-1} \alpha_k^t \left( \frac{\alpha_k^t L + \omega_1 + \omega_2}{2} U + \frac{L^2}{\omega_1} UK(1 + \log K)(\alpha_0^t)^2 + \frac{\sigma^2}{2\omega_2 b} \right) \\
&= \frac{LU}{2} \sum_{k=0}^{K-1} (\alpha_k^t)^2 + \left( \frac{L^2}{\omega_1} UK(1 + \log K)(\alpha_0^t)^2 + \frac{\omega_1 + \omega_2}{2} U + \frac{\sigma^2}{2\omega_2 b} \right) \sum_{k=0}^{K-1} \alpha_k^t.
\end{aligned}$$

Letting  $\omega_1 = \frac{L\alpha_0^t}{\sqrt{K}}$ ,  $\omega_2 = \frac{\sigma}{\sqrt{Ub}}$  yields

$$\begin{aligned}
& \mathbb{E} [\nabla f(\mathbf{x}_t)^T \mathbf{d}_{t+1}] + \frac{\alpha_0^t \sqrt{K}}{2V} \mathbb{E} [\|\nabla f(\mathbf{x}_t)\|_2] \\
&= \frac{LU}{2} \sum_{k=0}^{K-1} (\alpha_k^t)^2 + \left( LU \left( \sqrt{K}(1 + \log K) + \frac{1}{2\sqrt{K}} \right) \alpha_0^t + \frac{\sqrt{U}\sigma}{\sqrt{b}} \right) \sum_{k=0}^{K-1} \alpha_k^t \\
&\stackrel{(42,43)}{\leq} \frac{LU}{2} (1 + \log K) (\alpha_0^t)^2 + \left( LU \left( \sqrt{K}(1 + \log K) + \frac{1}{2\sqrt{K}} \right) \alpha_0^t + \frac{\sqrt{U}\sigma}{\sqrt{b}} \right) 2\sqrt{K}\alpha_0^t \\
&= \sqrt{K}LU \left( \left( \frac{1}{2\sqrt{K}} + 2\sqrt{K} \right) (1 + \log K) + \frac{1}{\sqrt{K}} \right) (\alpha_0^t)^2 + \frac{\sqrt{U}\sigma}{\sqrt{b}} 2\sqrt{K}\alpha_0^t.
\end{aligned} \tag{41}$$

On the other hand, by the smoothness assumption, we have

$$\begin{aligned}
& \mathbb{E} [f(\mathbf{x}_{t+1}) - f(\mathbf{x}_t)] \leq \mathbb{E} [\nabla f(\mathbf{x}_t)^T \mathbf{d}_{t+1}] + \frac{L}{2} \mathbb{E} [\|\mathbf{d}_{t+1}\|^2] \\
&\stackrel{(41)}{\leq} -\frac{\alpha_0^t \sqrt{K}}{2V} \mathbb{E} [\|\nabla f(\mathbf{x}_t)\|_2] + \sqrt{K}LU \left( \left( \frac{1}{2\sqrt{K}} + 2\sqrt{K} \right) (1 + \log K) + \frac{1}{\sqrt{K}} \right) (\alpha_0^t)^2 + \frac{\sqrt{U}\sigma}{\sqrt{b}} 2\sqrt{K}\alpha_0^t + \frac{L}{2} \mathbb{E} [\|\mathbf{d}_{t+1}\|^2] \\
&\stackrel{(27)}{\leq} -\frac{\alpha_0^t \sqrt{K}}{2V} \mathbb{E} [\|\nabla f(\mathbf{x}_t)\|_2] + \underbrace{\sqrt{K}LU \left( \left( \frac{1}{2\sqrt{K}} + \frac{5}{2}\sqrt{K} \right) (1 + \log K) + \frac{1}{\sqrt{K}} \right) (\alpha_0^t)^2 + \frac{\sqrt{U}\sigma}{\sqrt{b}} 2\sqrt{K}\alpha_0^t}_{\triangleq \hat{\Psi}},
\end{aligned}$$

where in the last step we have reused the bound in Lemma 1. Summing the above up for  $t = 0, \dots, T-1$  gives

$$\begin{aligned}
\sum_{t=0}^{T-1} \alpha_0^t \frac{\mathbb{E} [\|\nabla f(\mathbf{x}_t)\|_2]}{V} &\leq 2 \frac{f(\mathbf{x}_0) - f_*}{\sqrt{K}} + 4 \frac{\sqrt{U}\sigma}{\sqrt{b}} \sum_{t=0}^{T-1} \alpha_0^t + 2LU\hat{\Psi} \sum_{t=0}^{T-1} (\alpha_0^t)^2 \\
&\stackrel{(43,45)}{\leq} 2 \frac{f(\mathbf{x}_0) - f_*}{\sqrt{K}} + \frac{16}{3} \frac{\sqrt{U}\sigma}{\sqrt{b}} \alpha T^{\frac{3}{4}} + 4LU\hat{\Psi}\alpha^2\sqrt{T} \\
&\stackrel{b \geq \sqrt{T}}{\leq} 2 \frac{f(\mathbf{x}_0) - f_*}{\sqrt{K}} + \frac{16}{3} \sqrt{U}\sigma\alpha\sqrt{T} + 4LU\hat{\Psi}\alpha^2\sqrt{T}.
\end{aligned}$$

The left-hand side is bounded from below as

$$\sum_{t=0}^{T-1} \alpha_0^t \frac{\mathbb{E} [\|\nabla f(\mathbf{x}_t)\|_2]}{V} \stackrel{(46)}{\geq} \alpha T^{\frac{3}{4}} \frac{1}{T} \sum_{t=0}^{T-1} \frac{\mathbb{E} [\|\nabla f(\mathbf{x}_t)\|_2]}{V}.$$

We therefore have

$$\frac{1}{T} \sum_{t=0}^{T-1} \frac{\mathbb{E} [\|\nabla f(\mathbf{x}_t)\|_2]}{V} \leq \frac{2}{T^{\frac{3}{4}}} \frac{f(\mathbf{x}_0) - f_*}{\alpha\sqrt{K}} + \left( \frac{16}{3}\sigma + 4L\sqrt{U}\hat{\Psi}\alpha \right) \frac{\sqrt{U}}{T^{\frac{1}{4}}}.$$

The final step is to specify the value of  $U$  which is an upper bound of  $\mathbb{E}[\|\mathbf{u}_{i,k}^t\|^2]$ . For any  $\mathbf{u}_{i,k}^t$  drawn from  $\mathcal{M}_l^G$ , we know from Proposition 1 that it has an identical covariance matrix and all its coordinates are independently distributed. It means Lemma 7 can be used here. In particular, since we are considering the setting  $\|\cdot\| = \|\cdot\|_2$ , we can choose  $U = n$ . Substituting the value of  $V$  in (40) into the above inequality completes the proof.  $\square$

*Proof of Theorem 5.* The proof is almost identical to that of Theorem 4, since by Propositions 1 and 2 the two mixture sampling schemes only differ in the fourth-order moment, which is used in bounding

$$\mathbb{E} [\|\nabla f(\mathbf{x}_t)^T \mathbf{u}_{i,k}^t\|]$$

in (39). Note that by Proposition 2 the above can be lower bounded by  $\frac{\|\nabla f(\mathbf{x}_t)\|_2}{\sqrt{n/(sl)+3}}$ . Therefore, we can simply replace the value of  $V$  in (40) by

$$V_t = \sqrt{3 + n/(sl)}$$

and we will get the final bound.  $\square$

## APPENDIX F

### AUXILIARY LEMMAS

**Lemma 7.** Let  $\|\cdot\|$  be a vector norm in  $\mathbb{R}^n$ . Let  $\mathbf{u} \in \mathbb{R}^n$  be any random vector satisfying  $\mathbb{E}[\mathbf{u}] = \mathbf{0}$  and  $\mathbb{V}[\mathbf{u}] = \mathbf{I}$ . Assume all coordinates of  $\mathbf{u}$  are distributed independently. Then, there exists a constant  $U > 0$  such that  $\mathbb{E}[\|\mathbf{u}\|^2] \leq U$ . In particular, we can choose  $U = n$  for  $\|\cdot\| = \|\cdot\|_2$  and  $U = 4 \log(\sqrt{2}n)$  for  $\|\cdot\| = \|\cdot\|_\infty$ .

*Proof.* First, by the identity covariance matrix assumption, we have  $\mathbb{E}[\|\mathbf{u}\|_2^2] = \text{Tr}[\mathbb{E}[\mathbf{u}\mathbf{u}^T]] = \text{Tr}[\mathbb{V}[\mathbf{u}]] = \text{Tr}[\mathbf{I}] = n$ , where  $\text{Tr}[\cdot]$  denotes the matrix trace. So this suggests the  $\ell_2$  norm of  $\mathbf{u}$  can be bounded by  $U \triangleq n$ . Then, due to the equivalence of vector norm, we know such a bound exists for all norms. In the next, we study the case of  $\ell_\infty$  norm.

Let  $t \in (0, 1/2)$  be a constant. By the convexity of  $\|\cdot\|_\infty$ , we have

$$\exp(t\mathbb{E}[\|\mathbf{u}\|_\infty^2]) \leq \mathbb{E}[\exp(t\|\mathbf{u}\|_\infty^2)] = \mathbb{E}\left[\exp\left(t \max_{1 \leq i \leq n} u_i^2\right)\right] \leq \sum_{i=1}^n \mathbb{E}[\exp(tu_i^2)] = n\mathbb{E}[\exp(tu_1^2)],$$

where the last equation is due to that all elements in  $\mathbf{u}$  are independently distributed.

The rightmost expectation can be calculated explicitly as

$$\mathbb{E}[\exp(tu_1^2)] = \frac{1}{\sqrt{2\pi}} \int \exp(tu_1^2) \exp\left(-\frac{1}{2}u_1^2\right) du_1 = \frac{1}{\sqrt{2\pi}} \int \exp\left(-\frac{1-2t}{2}u_1^2\right) du_1 = \frac{1}{\sqrt{1-2t}}.$$

It follows that

$$\exp(t\mathbb{E}[\|\mathbf{u}\|_\infty^2]) \leq \frac{n}{\sqrt{1-2t}}$$

and therefore

$$\mathbb{E}[\|\mathbf{u}\|_\infty^2] \leq \frac{1}{t} \log \frac{n}{\sqrt{1-2t}}.$$

Note that this inequality holds for any  $t \in (0, 1/2)$ . So we can choose  $t = 1/4$  and then the desired bound  $\mathbb{E}[\|\mathbf{u}\|^2] \leq 4 \log(\sqrt{2}n)$  follows.  $\square$

**Lemma 8.** For  $J \in \mathbb{Z}_+$  we have the following properties for the partial sum of  $p$ -series with  $p = 1, 0.5$ , or  $0.25$ :

$$\sum_{j=1}^J \frac{1}{j} \leq 1 + \log J \tag{42}$$

$$\sqrt{J} \leq \sum_{j=1}^J \frac{1}{j^{0.5}} \leq 2\sqrt{J} \tag{43}$$

$$\sum_{j=1}^J \frac{a_j}{j^{0.5}} \geq \sqrt{J} \left( \frac{1}{J} \sum_{j=1}^J a_j \right), \forall a_j \geq 0 \tag{44}$$

$$\sum_{j=1}^J \frac{1}{j^{0.25}} \leq \frac{4}{3} J^{3/4} \tag{45}$$

$$\sum_{j=1}^J \frac{a_j}{j^{0.25}} \geq J^{3/4} \left( \frac{1}{J} \sum_{j=1}^J a_j \right), \forall a_j \geq 0 \tag{46}$$

*Proof.* (44) and (46) are trivial. For (42), (43) and (45) see [56, Section 4.1].  $\square$

**Lemma 9.** For  $\beta < \sqrt{\frac{1}{2\sqrt{2}}}$ , we have the following bounds

$$\sum_{j=1}^t \frac{\beta^{t-j}}{j^{0.25}} = \frac{\beta^{t-1}}{1^{0.25}} + \cdots + \frac{\beta^0}{t^{0.25}} \leq \frac{20}{t^{0.25}}, \tag{47}$$

$$\sum_{j=1}^t \frac{(2\beta^2)^{t-j}}{\sqrt{j}} = \frac{(2\beta^2)^{t-1}}{\sqrt{1}} + \cdots + \frac{(2\beta^2)^0}{\sqrt{t}} \leq \frac{1}{\sqrt{t}(1 - 2\sqrt{2}\beta^2)} \tag{48}$$

*Proof.* Both bounds hold trivially for  $t = 1$ , so we prove them with induction.

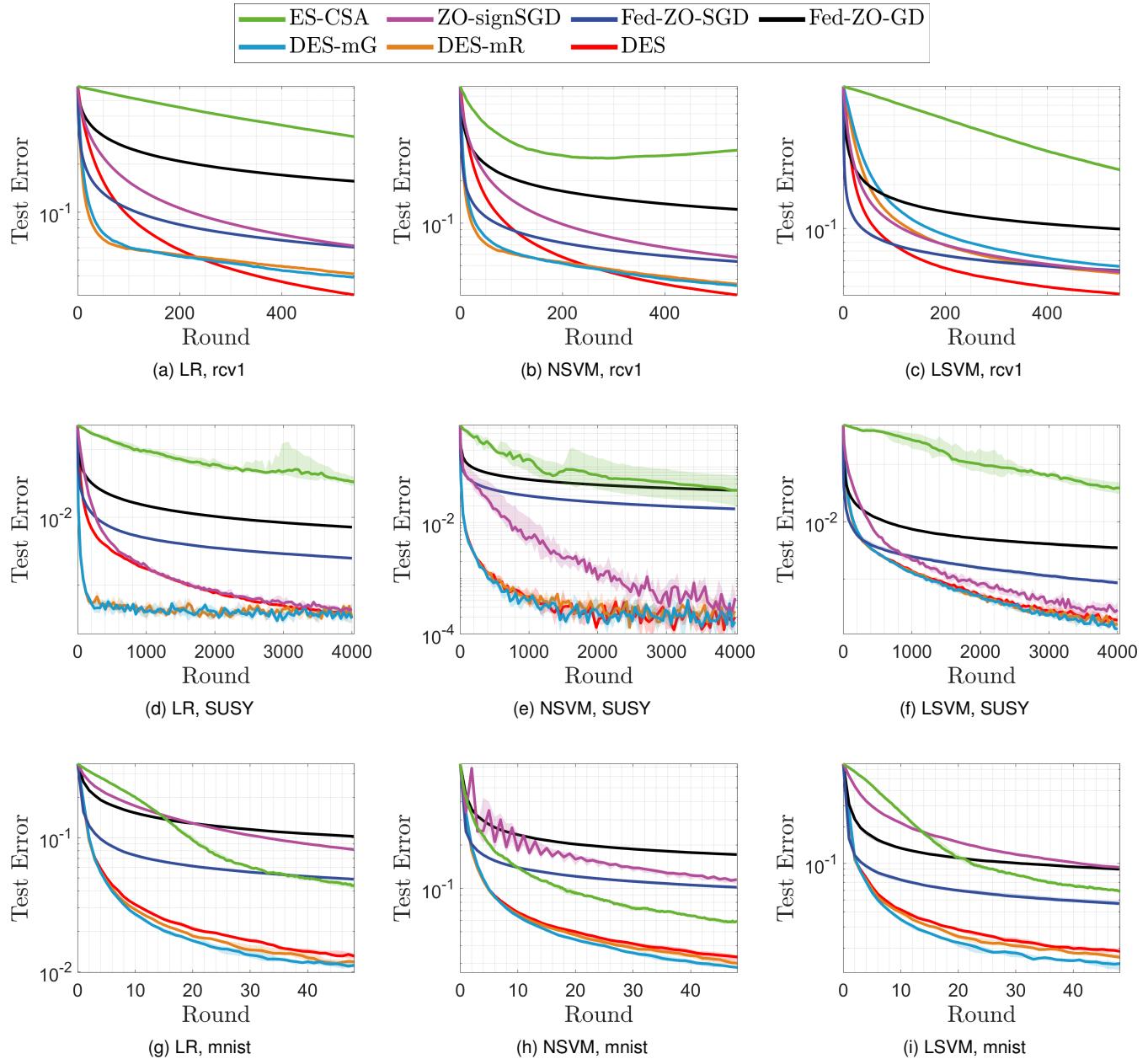


Fig. 7. Generalization performance on rcv1, SUSY, and mnist datasets. The curve displays the test error versus the number of rounds and the corresponding shaded area extends from the 25th to 75th percentiles over the results obtained from all independent runs.

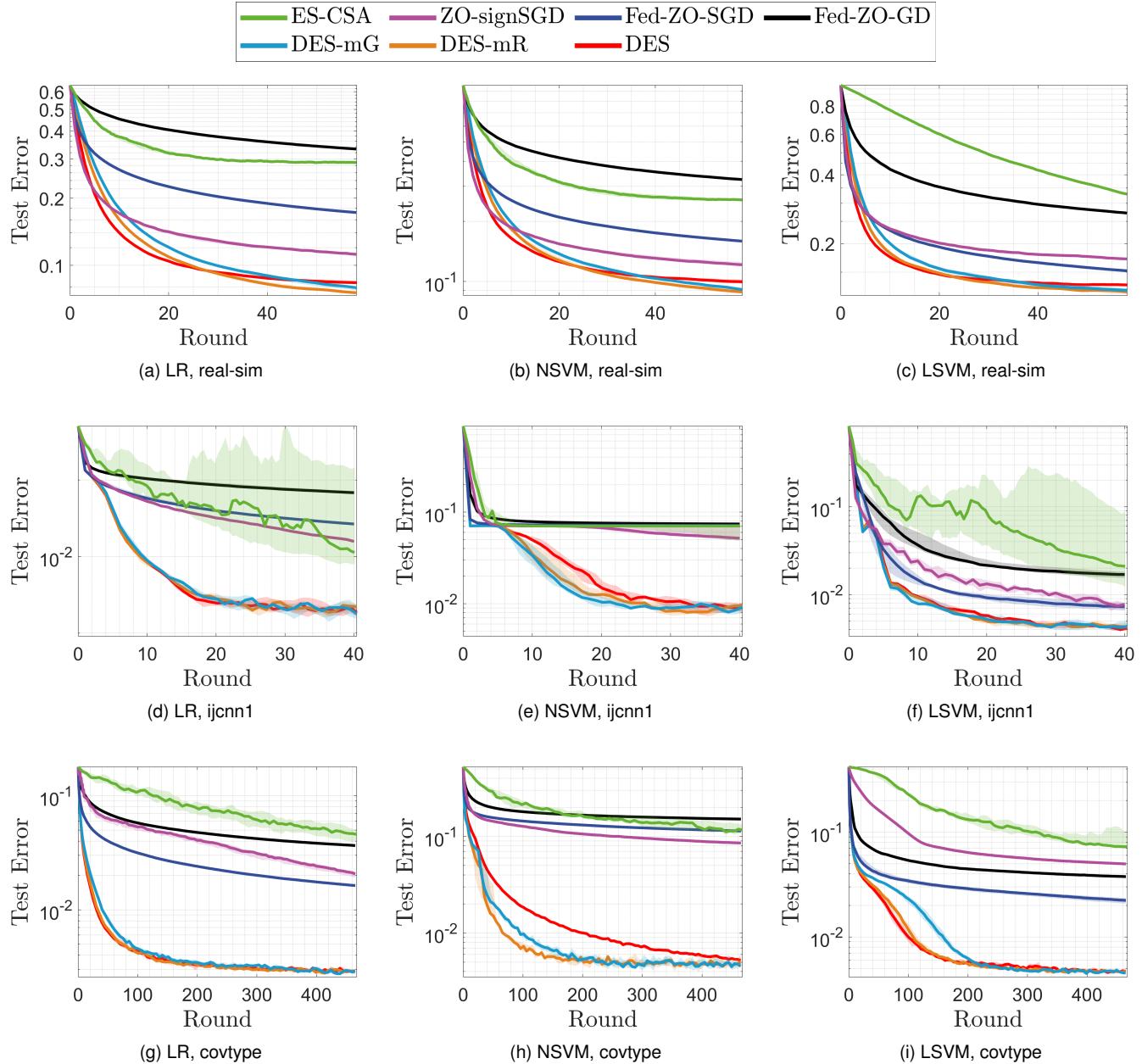


Fig. 8. Generalization performance on real-sim, ijCNN1, and covtype datasets. The curve displays the test error versus the number of rounds and the corresponding shaded area extends from the 25th to 75th percentiles over the results obtained from all independent runs.