Quantization Avoids Saddle Points in Distributed Optimization

Yanan Bo[a] **and Yongqiang Wang**[a,1]

This manuscript was compiled on March 14, 2024

Distributed nonconvex optimization underpins key functionalities of numerous distributed systems, ranging from power systems, smart buildings, cooperative robots, vehicle networks to sensor networks. Recently, it has also merged as a promising solution to handle the enormous growth in data and model sizes in deep learning. A fundamental problem in distributed nonconvex optimization is avoiding convergence to saddle points, which significantly degrade optimization accuracy. We discover that the process of quantization, which is necessary for all digital communications, can be exploited to enable saddle-point avoidance. More specifically, we propose a stochastic quantization scheme and prove that it can effectively escape saddle points and ensure convergence to a second-order stationary point in distributed nonconvex optimization. With an easily adjustable quantization granularity, the approach allows a user to control the number of bits sent per iteration and, hence, to aggressively reduce the communication overhead. Numerical experimental results using distributed optimization and learning problems on benchmark datasets confirm the effectiveness of the approach.

Quantization | Saddle-point Avoidance | Distributed Nonconvex Optimization

With the unprecedented advances in embedded electronics and communication technologies, cooperation or coordination has emerged as a key feature in numerous engineered systems such as smart grids, intelligent transportation systems, cooperative robots, cloud computing, and smart cities. This has spurred the development of distributed algorithms in which spatially distributed computing devices (hereafter referred to as agents), communicating over a network, cooperatively solve a task without resorting to a central coordinator/mediator that aggregates all data in the network. Many of these distributed algorithms boil down to the following distributed optimization problem:

$$\min_{\boldsymbol{\theta}\in\mathbb{R}^d} F(\boldsymbol{\theta}) = \frac{1}{N}\sum_{i=1}^{N} f_i(\boldsymbol{\theta}), \qquad [1]$$

where $f_i(\cdot) : \mathbb{R}^d \to \mathbb{R}$ denotes the local objective function private to agent i, $F(\cdot) : \mathbb{R}^d \to \mathbb{R}$ is the global objective function representing the network-level cost to be minimized cooperatively by all participating agents, and N is the number of agents.

Initially introduced in the 1980s in the context of parallel and distributed computation (1), the above distributed optimization problem has received intensive interest in the past decade due to the surge of smart systems and deep learning applications (2, 3). So far, plenty of approaches have been proposed to solve the above distributed optimization problem, with some of the commonly used approaches including gradient methods (see, e.g., (2, 4–8)), distributed alternating direction method of multipliers (see, e.g., (9)), and distributed Newton methods (see, e.g., (10)).

However, most of these approaches focus on convex objective functions, whereas results are relatively sparse for nonconvex objective functions. In fact, in many applications, the objective functions are essentially nonconvex. For example, in the resource allocation problem of communication networks, the utility functions are nonconvex when the communication traffic is non-elastic (11); in most machine learning applications, the objective functions are nonconvex due to the presence of multi-layer neural networks (12); in policy optimization for linear-quadratic regulators (13) as well as for robust and risk-sensitive control (14), nonconvex optimization naturally arises.

In nonconvex optimization, oftentimes, the most fundamental problem is to avoid saddle points (stationary points that are not local extrema). For

Significance Statement

Distributed optimization underpins key functionalities of numerous engineered systems such as smart grids, intelligent transportation, and smart cities. It is also reshaping the landscape of machine learning due to its inherent advantages in handling large data/model sizes. However, saddle-point avoidance becomes extremely challenging in distributed optimization because individual agents in distributed optimization do not have access to the global gradient. We show that quantization effects, which are unavoidable due to communications in distributed optimization and regarded as detrimental in existing studies, can be exploited to enable saddle-point avoidance for free. By judiciously designing the quantization scheme, we propose an approach that evades saddle points and ensures convergence to a second-order stationary point in distributed nonconvex optimization.

Author affiliations: [a]Department of Electrical and Computer Engineering, Clemson University, Clemson, SC 29634 USA

Y.B and Y.W designed research, performed research and wrote the paper.

The authors declare no competing interest.

[1]To whom correspondence should be addressed. E-mail: yongqiw@clemson.edu

example, in machine learning applications, it has been shown that the main bottleneck in parameter optimization is not due to the existence of multiple local minima but the existence of many saddle points that trap gradient updates (15). The problem of saddle points is more acute in deep neural networks, where saddle points are usually encircled by high-error plateaus, exerting substantial deceleration on the learning process while engendering a deceptive semblance of the presence of a local minimum (16, 17). To escape saddle points, classical approaches resort to second-order information, in particular, the Hessian matrix of second-order derivatives (see, e.g., (18, 19)). The Hessian matrix-based approach, however, incurs high costs in both computation and storage. This is because the dimension of the Hessian matrix increases quadratically with an increase in the optimization-variable dimension, which can scale to hundreds of millions in modern deep learning applications (20). Recently, random perturbations of first-order gradient methods have been shown capable of escaping saddle points in centralized optimization (see, e.g., (15, 21)). However, it is unclear if this is still true in decentralized nonconvex optimization, where the decentralized architecture brings in fundamental differences in optimization dynamics. For example, in decentralized optimization, the saddle points of individual local objective functions $f_i(\cdot)$ are different from those of the global objective function $F(\cdot)$, which is the only function that needs to be considered in centralized optimization. In fact, in distributed optimization, all local objective functions $f_i(\cdot)$ are private to individual agents, preventing any single agent from accessing the global objective function $F(\cdot)$ and further from exploiting the gradient/Hessian information of $F(\cdot)$ in its local iteration to avoid the saddle points of $F(\cdot)$. In addition, the inter-agent coupling also complicates the optimization dynamics. Note that random algorithm initialization has been shown to be able to asymptotically avoid saddle points in centralized nonconvex optimization (22), which has been further extended to the decentralized case in (23). However, the result in (21) shows that this approach to avoiding saddle points may take an exponentially longer time, rendering it impractical.

In this paper, we propose to exploit the effects of quantization, which are naturally inherent to all digital communication methods, to evade saddle points in distributed nonconvex optimization. The process of quantization is necessary in all modern communications to represent continuous-valued variables with a smaller set of discrete-valued variables since digital communication channels can only transmit/receive bit streams. The conversion from continuous-valued variables to discrete-valued variables inevitably leads to rounding and truncation errors. In fact, in distributed learning for deep neural networks, since model parameters or gradients have to be shared across agents in every iteration and the dimension of these model parameters and gradients can easily scale to hundreds of millions (20), it is a common practice to use coarse quantization schemes or compression techniques to reduce the overhead of communication (24, 25). Recently, plenty of distributed optimization and learning algorithms have been proposed that can ensure provable convergence to the optimal solution in the convex case (see, e.g., (26–34)) or to first-order stationary points in the nonconvex case (see, e.g., (35–37)), even in the presence quantization/compression errors. However, in all these existing results, quantization effects are treated as detrimental to the distributed optimization process and have to be suppressed to ensure convergence accuracy. In this paper, to the contrary, we exploit quantization effects to evade saddle points and hence improve convergence accuracy in distributed nonconvex optimization. By judiciously designing the quantization scheme, we propose an algorithm that can make use of quantization effects to effectively escape saddle points and ensure convergence to second-order stationary points. To the best of our knowledge, this is the first time that quantization is shown to be beneficial to the convergence accuracy of distributed optimization. The proposed quantization scheme can also aggressively reduce the overhead of communication, which is widely regarded as the bottleneck in distributed training of machine-learning models (24).

Problem Formulation

Notations. We use bold letters to denote matrices and vectors, i.e., $\boldsymbol{A}$ and $\boldsymbol{x}$. We use $\|\cdot\|$ to represent the ℓ_2 norm of vectors and the Frobenius norm of matrices. For a function $F(\cdot) : \mathbb{R}^d \to \mathbb{R}$, we use $\nabla F(\cdot)$ and $\nabla^2 F(\cdot)$ to denote its gradient and Hessian, respectively. We use $\mathcal{O}(\cdot)$ to hide absolute constants that do not depend on any problem parameter. We use $[N]$ to represent the set $\{1, 2, \cdots, N\}$. We use $\lambda_{\min}(\cdot)$ to represent the minimal eigenvalue of a matrix.

Formulation. We consider a distributed optimization problem where N agents, each with its own local objective function, collaboratively optimize the network-level sum (average) of all local objective functions. Since the local objective functions are private to individual agents, no agents have access to the global objective function. To solve the distributed optimization problem, individual agents have to share local intermediate optimization variables with their respective immediate neighboring agents to ensure convergence to a desired solution. We describe the local interaction among agents using a weight matrix $\boldsymbol{A} = [a_{ij}]_{N\times N}$, where $a_{ij} > 0$ if agent j and agent i can directly communicate with each other, and $a_{ij} = 0$ otherwise. For an agent $i \in [N]$, its neighbor set $\mathcal{N}_i$ is defined as the collection of agents j such that $a_{ij} > 0$. a_{ii} represents self-interaction, i.e., the influence of agent i's optimization variable at iteration k on its optimization variable at iteration $k+1$. Furthermore, we make the following assumption on $\boldsymbol{A}$:

Assumption 1. *The matrix $\boldsymbol{A} = \{a_{ij}\} \in \mathbb{R}^{N\times N}$ is symmetric and satisfies $\mathbf{1}^\top \boldsymbol{A} = \mathbf{1}^\top$, $\boldsymbol{A}\mathbf{1} = \mathbf{1}$, and $\|\boldsymbol{A} - \frac{\mathbf{1}\mathbf{1}^\top}{N}\| < 1$.*

Assumption 1 ensures that the interaction graph induced by $\boldsymbol{A}$ is balanced and connected, i.e., there is a path from each agent to every other agent.

The optimization problem in [1] can be reformulated as the following multi-agent optimization problem:

$$\begin{aligned} \min_{\boldsymbol{x}\in\mathbb{R}^{N\times d}} \; & f(\boldsymbol{x}) = \frac{1}{N}\sum_{i=1}^{N} f_i(\boldsymbol{x}_i), \\ \text{s.t.} \quad & \boldsymbol{x}_1 = \boldsymbol{x}_2 = \cdots = \boldsymbol{x}_N, \end{aligned} \qquad [2]$$

where the matrix $\boldsymbol{x}$ is composed of all the local optimization variables, i.e., $\boldsymbol{x} = \left[\boldsymbol{x}_1^\top; \boldsymbol{x}_2^\top; \ldots; \boldsymbol{x}_N^\top\right] \in \mathbb{R}^{N\times d}$.
In this paper, the local objective function $f_i(\boldsymbol{x}_i)$ and global

objective function $f(\boldsymbol{x})$ can be nonconvex. They are assumed to satisfy the following conditions:

Assumption 2. *Every $f_i(\cdot)$ is differentiable and is L_i-Lipschitz as well as ρ_i-Hessian Lipschitz:*

$$\|\nabla f_i(\boldsymbol{x_1}) - \nabla f_i(\boldsymbol{x_2})\| \leqslant L_i \|\boldsymbol{x_1} - \boldsymbol{x_2}\|, \quad \forall \boldsymbol{x_1}, \boldsymbol{x_2} \in \mathbb{R}^d, \quad [3]$$

$$\|\nabla^2 f_i(\boldsymbol{x_1}) - \nabla^2 f_i(\boldsymbol{x_2})\| \leqslant \rho_i \|\boldsymbol{x_1} - \boldsymbol{x_2}\|, \quad \forall \boldsymbol{x_1}, \boldsymbol{x_2} \in \mathbb{R}^d. \quad [4]$$

It can be verified that the global gradient $\nabla F(\boldsymbol{\theta}) = \frac{1}{N}\sum_{i=1}^{N} \nabla f_i(\boldsymbol{\theta})$ and Hessian $\nabla^2 F(\boldsymbol{\theta}) = \frac{1}{N}\sum_{i=1}^{N} \nabla^2 f_i(\boldsymbol{\theta})$ are L-Lipschitz and ρ-Hessian Lipschitz, with $L = \frac{1}{N}\sum_i L_i$ and $\rho = \frac{1}{N}\sum_i \rho_i$.

As in most existing results on distributed nonconvex optimization, we assume that the local gradients $\nabla f_i(\cdot)$ are bounded:

Assumption 3. *There exists a constant G such that $\|\nabla f_i(\boldsymbol{\theta})\| \leqslant G$ holds for all $\boldsymbol{\theta} \in \mathbb{R}^d$ and $i \in [N]$.*

In this paper, we will show that quantization can help evade saddle points and ensure convergence to second-order stationary points in distributed nonconvex optimization. To this end, we first recall the following definitions for first-order stationary points, saddle points, and second-order stationary points, which are commonly used in the study of saddle-point problems:

Definition 1. *For a twice differentiable objective function $F(\cdot)$, we call $\boldsymbol{\theta}^\star \in \mathbb{R}^d$ a first-order (respt. second-order) stationary point if $\nabla F(\boldsymbol{\theta}^\star) = \mathbf{0}$ (respt. $\nabla F(\boldsymbol{\theta}^\star) = \mathbf{0}$ and $\lambda_{\min}(\nabla^2 F(\boldsymbol{\theta}^\star)) \geqslant 0$) holds. Moreover, a first-order stationary point $\boldsymbol{\theta}^\star$ can be viewed as belonging to one of the three categories:*

- local minimum*: there exists a scalar $\gamma > 0$ such that $F(\boldsymbol{\theta}^\star) \leqslant F(\boldsymbol{\theta})$ holds for any $\boldsymbol{\theta}$ satisfying $\|\boldsymbol{\theta}^\star - \boldsymbol{\theta}\| \leqslant \gamma$;*
- local maximum*: there exists a scalar $\gamma > 0$ such that $F(\boldsymbol{\theta}^\star) \geqslant F(\boldsymbol{\theta})$ holds for any $\boldsymbol{\theta}$ satisfying $\|\boldsymbol{\theta}^\star - \boldsymbol{\theta}\| \leqslant \gamma$;*
- saddle point*: neither of the above two cases is true, i.e., for any scalar $\gamma > 0$, there exist $\boldsymbol{\theta_1}$ and $\boldsymbol{\theta_2}$ satisfying $\|\boldsymbol{\theta_1} - \boldsymbol{\theta}^\star\| \leqslant \gamma$ and $\|\boldsymbol{\theta_2} - \boldsymbol{\theta}^\star\| \leqslant \gamma$ such that $F(\boldsymbol{\theta_1}) < F(\boldsymbol{\theta}^\star) < F(\boldsymbol{\theta_2})$ holds.*

Since distinguishing saddle points from local minima for smooth functions is NP-hard in general (38), we focus on a subclass of saddle points, i.e., $\epsilon-$strict saddle points:

Definition 2. ($\epsilon-$strict saddle point and $\epsilon-$second-order stationary point) *For a twice-differentiable function $F(\cdot)$, we say that $\boldsymbol{\theta}^\star \in \mathbb{R}^d$ is an $\epsilon-$strict saddle point if 1) $\boldsymbol{\theta}^\star$ is an $\epsilon-$first-order stationary point i.e., $\|\nabla F(\boldsymbol{\theta}^\star)\| \leqslant \epsilon$; and 2) $\lambda_{\min}(\nabla^2 F(\boldsymbol{\theta}^\star)) \leqslant -\sqrt{\rho\epsilon}$, where ρ is the Hessian Lipschitz parameter in Assumption 2. Similarly, $\boldsymbol{\theta}^\star \in \mathbb{R}^d$ is an $\epsilon-$second-order stationary point if 1) $\boldsymbol{\theta}^\star$ is an $\epsilon-$first-order stationary point, i.e., $\|\nabla F(\boldsymbol{\theta}^\star)\| \leqslant \epsilon$ and 2) $\lambda_{\min}(\nabla^2 F(\boldsymbol{\theta}^\star)) > -\sqrt{\rho\epsilon}$.*

For a smooth function, a generic saddle point must satisfy that the minimum eigenvalue of its Hessian is non-positive. Our consideration of strict saddle points rules out the case where the minimum eigenvalue of the Hessian is zero. A line of recent work in the machine learning literature shows that for many popular models in machine learning, all saddle points are indeed strict saddle points, with examples ranging from tensor decomposition (15), dictionary learning (39), smooth semidefinite programs (40), to robust principal component analysis (41).

Proposed Algorithm

By exploiting the effects of quantization, we propose a distributed nonconvex optimization algorithm that can ensure the avoidance of saddle points and convergence to a second-order stationary point. The detailed algorithm is summarized in Algorithm 1.

Algorithm 1 Distributed Optimization with Guaranteed Saddle-point Avoidance

```
Initialization: $\boldsymbol{x}_i^0 \in \mathbb{R}^d$ for every agent $i$;
Parameters: Stepsize sequences $\{\varepsilon_k\}$ and $\{\eta_k\}$;
            Quantization level $\ell$;
for $k = 1, 2, ...$ do
    for all $i \in [N]$ do
        1. Quantize its decision vector $\boldsymbol{x}_i^k$ to obtain $Q_\ell(\boldsymbol{x}_i^k)$
           and send the quantized $Q_\ell(\boldsymbol{x}_i^k)$ to all neighbor agents in $\mathcal{N}_i$;
        2. Receive $Q_\ell(\boldsymbol{x}_j^k)$ from neighbor agents $j \in \mathcal{N}_i$ and calculate
           the following estimate of the global optimization variable:
           $\tilde{\boldsymbol{x}}_i^{k+1} = \boldsymbol{x}_i^k + \varepsilon_k \sum_{j \in \mathcal{N}_i \cup \{i\}} a_{ij}(Q_\ell(\boldsymbol{x}_j^k) - \boldsymbol{x}_i^k);$   [5]
        3. Calculate local gradient $\nabla f_i(\boldsymbol{x}_i^k)$ and update $\boldsymbol{x}_i^{k+1}$ by:
           $\boldsymbol{x}_i^{k+1} = \tilde{\boldsymbol{x}}_i^{k+1} - \eta_k \nabla f_i(\boldsymbol{x}_i^k).$   [6]
    end for
end for
```

As key components of our approach to evading saddle points and ensuring convergence accuracy, we propose the following quantization scheme and stepsize strategy:

Quantization Scheme. Our quantization scheme is inspired by the QSGD quantization scheme proposed in (25) and the TernGrad quantization scheme in (24). (Note that the QSGD and TernGrad schemes were proposed to quantize gradients, whereas our Algorithm 1 quantizes optimization variables.) More specifically, at each time instant, we represent a continuous-valued variable with a randomized rounding to a set of quantization points with adjustable discrete quantization levels in a way that preserves the statistical properties of the original. However, different from (25), to ensure saddle-point avoidance, we employ two sets of quantization levels and purposely switch between the two sets of quantization levels

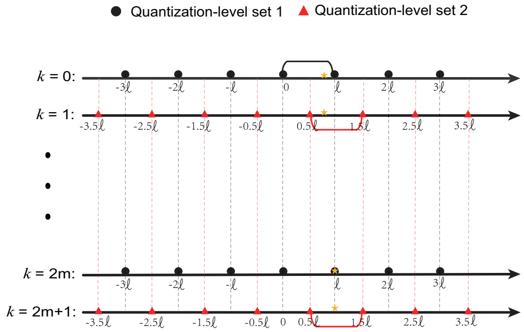

Fig. 1. The proposed quantization scheme with quantization interval ℓ. The star represents a value to be quantized, and it is located in the quantization interval of $[0, \ell]$ under level-set 1 and $[0.5\ell, 1.5\ell]$ under level-set 2. At any even-number iteration (k is even), the star value will be quantized to either 0 or ℓ, with respective probabilities provided in [7]. At any odd-number iteration (k is odd), the star value will be quantized to either 0.5ℓ or 1.5ℓ, with respective probabilities given in [8].

in a periodic manner. The detailed scheme is described below:

For any $\boldsymbol{v} = [v_1, v_2, \ldots, v_d] \in \mathbb{R}^d$,

1. At any even-number iteration (k is even), map every $v_i \in \mathbb{R}$ onto the quantization level-set: $\{\cdots, -3\ell, -2\ell, -\ell, 0, \ell, 2\ell, 3\ell, \cdots\}$ (which we will refer to "level-set 1" hereafter) as follows:

$$Q_\ell(v_i) = \begin{cases} n\ell, & \text{with probability } 1 - p(v_i, \ell) \\ (n+1)\ell, & \text{with probability } p(v_i, \ell) \end{cases} \qquad [7]$$

where $n \in \mathbb{Z}$ is determined by the inequality $n\ell \leqslant v_i < (n+1)\ell$, and the probability $p(v_i, \ell)$ is given by $p(v_i, \ell) = \frac{v_i}{\ell} - n$.

2. At any odd-number iteration (k is odd), map every $v_i \in \mathbb{R}$ onto the quantization level-set: $\{\cdots, -2.5\ell, -1.5\ell, -0.5\ell, 0.5\ell, 1.5\ell, 2.5\ell, \cdots\}$ (which we will refer to "level-set 2" hereafter) as follows:

$$Q_\ell(v_i) = \begin{cases} (n' - 0.5)\ell, & \text{with probability } 1 - p'(v_i, \ell) \\ (n' + 0.5)\ell, & \text{with probability } p'(v_i, \ell) \end{cases} \qquad [8]$$

where $n' \in \mathbb{Z}$ is determined by the inequality $(n' - 0.5)\ell \leqslant v_i < (n' + 0.5)\ell$, and the probability $p'(v_i, \ell)$ is given by $p'(v_i, \ell) = \frac{v_i}{\ell} - n' + 0.5$.

It is worth noting that compared with existing quantization schemes such as (25), this periodic switching between two sets of quantization levels does not introduce extra communication overheads. However, it avoids the possibility that any quantization input v_i always coincides with an endpoint of a quantization interval, resulting in a deterministic quantization output. Namely, for any point $\boldsymbol{v} \in \mathbb{R}^d$, it gives two different representations in the quantized space, which is key to perturb and avoid the state from staying on undesired saddle points under a non-zero stepsize (which will be elaborated later).

An instantiation of this quantization scheme is depicted in Fig. 1. It can be verified that the proposed quantization scheme satisfies the following properties:

Lemma 1. *For any $\boldsymbol{v} \in \mathbb{R}^d$, our quantization scheme $Q_\ell(\boldsymbol{v}) = [Q_\ell(v_1), Q_\ell(v_2), ..., Q_\ell(v_d)]$ has the following properties:*

1. *Unbiased quantization:* $\mathbb{E}[Q_\ell(\boldsymbol{v})] = \boldsymbol{v}$,
2. *Bounded variance:* $\mathbb{E}\left[\|Q_\ell(\boldsymbol{v}) - \boldsymbol{v}\|^2\right] \leqslant d\ell^2$.

Stepsize Strategy. In addition to purposely employing switching in the quantization scheme, the stepsizes $\{\varepsilon_k\}$ and $\{\eta_k\}$ in Algorithm 1 also have to be judiciously designed so as to evade saddle points and ensure convergence to a second-order stationary point. Intuitively speaking, in the early stage where saddle points may trap the optimization process, the stepsize ε_k should be large enough to ensure that the switching quantization-induced perturbation can effectively stir the evolution of optimization variables. However, to ensure that the optimization process can converge to an optimal solution, the quantization effect should gradually diminish, or in other words, ε_k should converge to zero. In addition, in distributed optimization, to ensure that all agents can converge to an optimal solution without any error, the stepsize η_k also has to converge to zero (different from the centralized case, in distributed optimization, a constant stepsize will lead to optimization errors that are in the order of the stepsize (2, 42, 43)). Moreover, to ensure that all agents can converge to the same optimal solution, the stepsize ε_k should decay slower than η_k (44–47). To fulfill these requirements, we design the stepsize sequences $\{\varepsilon_k\}$ and $\{\eta_k\}$ as follows:

1. Choose two positive constants α and β sequentially that satisfy the following relations: $0.6 < \alpha < \frac{2}{3}$ and $\frac{3}{2}\alpha < \beta < 1$. And then use these constants to construct two reference functions $\frac{c_1}{1+c_2 t^\alpha}$ and $\frac{c_1}{1+c_2 t^\beta}$, where t is continuous time and c_1 and c_2 are all positive constants.

2. For any probability p (where $1-p$ represents the desired probability of converging to a second-order stationary point, which can be chosen to be arbitrarily close to one, see the statement of Theorem 4 for details) and $\epsilon > 0$ given in Definition 2, select: $t_0 \geqslant \max\{C_1, C_2, C_3\}$, $t_{i+1} = t_i + \lceil \frac{1+c_2 t_i^\alpha}{c_1\sqrt{\rho\epsilon}} \rceil$ for $1 \leqslant i \leqslant I$, where $I = 30\max\{\frac{f_0 - f^\star}{Q}, \frac{2(f_0 - f^\star)\varepsilon_{t_0}}{\epsilon^2 \eta_{t_0}}\}$.[*]

3. The sequences $\{\varepsilon_k\}$ and $\{\eta_k\}$ for $\forall k \in \mathbb{Z}^+$ are given as follows:

$$\varepsilon_k = \begin{cases} \frac{c_1}{1+c_2 k^\alpha}, & k < t_0 \\ \frac{c_1}{1+c_2 t_i^\alpha}, & t_i \leqslant k < t_{i+1} \\ \frac{c_1}{1+c_2 k^\alpha}, & k \geqslant t_I \end{cases} \qquad [9]$$

$$\eta_k = \begin{cases} \frac{c_1}{1+c_2 k^\beta}, & k < t_0 \\ \frac{c_1}{1+c_2 t_i^\beta}, & t_i \leqslant k < t_{i+1} \\ \frac{c_1}{1+c_2 k^\beta}, & k \geqslant t_I \end{cases} \qquad [10]$$

[*] $C_1 = (\frac{4c_1^{2/3}(d_1+d_2)}{pc_2^{2/3}(1-\sigma_2)})^{\frac{3}{2\alpha}}$, $C_2 = (\frac{4(f_0-f^\star)(d_1+d_2)^{2/3}(1-\sigma_2)^{2/3}c_1}{c_2 p\epsilon^2\sqrt{\rho\epsilon}})^{\frac{1}{2\alpha-\beta}}$, $C_3 = (\frac{12\rho(d_1+d_2)^{1/6}}{(1-\sigma_2)^{1/6}\sqrt{\gamma}(\rho\epsilon)^{1/4}\ell})^{\frac{1}{\beta-4\alpha/3}}$, $Q = \frac{1}{60^2}\sqrt{\frac{\epsilon^3}{\rho}}$, where $d_1 = \frac{1+(1-\sigma_2)\varepsilon_0}{1-\sigma_2}G^2$, $d_2 = (1 + (1-\sigma_2)\varepsilon_0){\sigma_2}^2 Nd\ell^2$, and σ_2 is the second largest eigenvalue of $\boldsymbol{A}$. f_0 is the objective function value at $k = 0$. $f^\star$ denotes an estimated lower bound on the minimum global objective function $f(\cdot)$. For instance, in the matrix factorization problem where a low-rank matrix $U \in \mathbb{R}^{d\times r}$ is used to approximate a high-dimension matrix $\boldsymbol{M}^\star \in \mathbb{R}^{d\times d}$, the objective function is $f(\boldsymbol{U}) = \frac{1}{2}\|\boldsymbol{U}\boldsymbol{U}^\top - \boldsymbol{M}^\star\|_F^2$ and we can use $f^\star = 0$ as the lower bound (48).

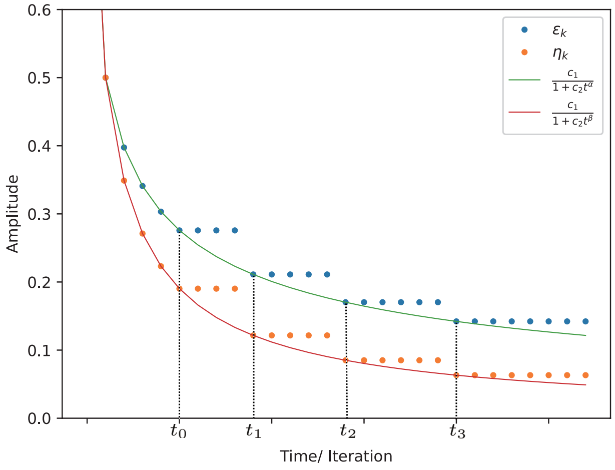

Fig. 2. An illustrative example of the stepsizes. The two solid curves represent two reference functions which are defined on the continuous time t. The blue and orange dots represent the values of stepsizes ε_k and η_k at discrete time instants k (which are periodic samples of the continuous time t). The time instants t_0, t_1, t_2, t_3 are determined in the second step of the stepsize strategy. Before t_0, the descent of the stepsize sequences is aligned with the reference functions. In intervals $[t_i, t_{i+1})$, the stepsizes remain constant, as described in the third step of the stepsize strategy.

The setup in [9] and [10] makes the stepsizes ε_k and η_k follow a decrease-and-hold pattern, as illustrated in Fig. 2. The rationale for this design can be understood intuitively as follows: To ensure that all agents can converge to a desired optimal solution, the iteration process must fulfill two objectives simultaneously: 1) ensure the consensual convergence of all agents to a stationary point; and 2) avoid saddle points. The "decrease" stages are important to fulfill the first objective in that the stepsizes ε_k and η_k are used to attenuate the quantization error and input heterogeneity among the agents, respectively, (both of which act as counter-forces for reaching consensus among all agents' iterates), and hence their decrease is key to ensure reaching consensus among the agents. The "hold" stages are necessary to accumulate enough stochastic quantization effects to stir the evolution of optimization variables and ensure saddle-point avoidance. It is worth noting that the decrease of stepsize η_k should also be carefully designed in both decreasing speed and timespan to ensure that sufficient gradient descent can be carried out to explore the solution space and ensure convergence to a stationary point. Hence we judiciously design the stepsize strategy to strike a balance between accumulating quantization noise to evade saddle points and attenuating quantization noise to ensure consensual convergence of all agents to a desired stationary point.

Based on the proposed quantization scheme and stepsize strategy, we can prove that the proposed Algorithm 1 can ensure all agents to evade saddle points and converge to the same second-order stationary point. For convenience of exposition, we divide the convergence analysis into two parts: "Consensual convergence to a first-order stationary point" and "Escaping saddle points and converging to a second-order stationary point". We leave all proofs in the Supporting Information.

Consensual Convergence to a First-order Stationary Point

We first prove that the proposed algorithm can ensure all agents to reach consensus on their optimization variables. For the convenience of analysis, we represent the effect of quantifying $\boldsymbol{x}_i^k$ as adding noise to $\boldsymbol{x}_i^k$, i.e., $Q_\ell(\boldsymbol{x}_i^k) = \boldsymbol{x}_i^k + \boldsymbol{\xi}_i^k$, where $\boldsymbol{\xi}_i^k$ is the stochastic quantization error. Using the iteration dynamics in [5] and [6], we can obtain the following relationship:

$$\boldsymbol{x}_i^{k+1} = (1-\varepsilon_k)\boldsymbol{x}_i^k + \varepsilon_k \boldsymbol{A} Q_\ell(\boldsymbol{x}_i^k) - \eta_k \nabla f_i(\boldsymbol{x}_i^k). \qquad [11]$$

By defining $\boldsymbol{x}^k = \left[(\boldsymbol{x}_1^k)^\top; (\boldsymbol{x}_2^k)^\top; \cdots; (\boldsymbol{x}_N^k)^\top\right] \in \mathbb{R}^{N\times d}$, $\boldsymbol{A}_k = (1-\varepsilon_k)\boldsymbol{I} + \varepsilon_k \boldsymbol{A}$, $\nabla f(\boldsymbol{x}^k) = \left[\nabla f_1^\top(\boldsymbol{x}_1^k); \nabla f_2^\top(\boldsymbol{x}_2^k); \cdots; \nabla f_N^\top(\boldsymbol{x}_N^k)\right] \in \mathbb{R}^{N\times d}$, and $\boldsymbol{\xi}^k = \left[(\boldsymbol{\xi}_1^k)^\top; (\boldsymbol{\xi}_2^k)^\top; \cdots; (\boldsymbol{\xi}_N^k)^\top\right] \in \mathbb{R}^{N\times d}$, we can recast the relationship in [11] into the following more compact form:

$$\boldsymbol{x}^{k+1} = \boldsymbol{A}_k \boldsymbol{x}^k + \varepsilon_k \boldsymbol{A}\boldsymbol{\xi}^k - \eta_k \nabla f(\boldsymbol{x}^k). \qquad [12]$$

Let $\bar{\boldsymbol{x}}^k$ be the average of all local optimization variables, i.e., $\bar{\boldsymbol{x}}^k = \frac{1}{N}\sum_{i=1}^N \boldsymbol{x}_i^k$. It can be verified that $\bar{\boldsymbol{x}}^k$ is equal to $\frac{(\boldsymbol{x}^k)^\top \boldsymbol{1}}{N}$, which can be further verified to satisfy the following relationship based on [12]:

$$\bar{\boldsymbol{x}}^{k+1} = \bar{\boldsymbol{x}}^k + \varepsilon_k \frac{(\boldsymbol{\xi}^k)^\top \boldsymbol{1}}{N} - \eta_k \frac{\nabla f^\top(\boldsymbol{x}^k)\boldsymbol{1}}{N}. \qquad [13]$$

Define the consensus error between individual agents' local optimization variables and the average optimization variable $\bar{\boldsymbol{x}}^k$ as $\boldsymbol{e}^k := \boldsymbol{x}^k - \boldsymbol{1}(\bar{\boldsymbol{x}}^k)^\top$. It can be verified that the i-th row of $\boldsymbol{e}^k$, i.e., $\boldsymbol{e}_i^k$, satisfies $\boldsymbol{e}_i^k = (\boldsymbol{x}_i^k)^\top - (\bar{\boldsymbol{x}}^k)^\top$. Using the algorithm iteration rule described in [5] and [6], we can obtain the following iteration dynamics for $\boldsymbol{e}^k$:

$$\boldsymbol{e}^{k+1} = \boldsymbol{A}_k \boldsymbol{e}^k + \varepsilon_k \boldsymbol{A}\boldsymbol{W}\boldsymbol{\xi}^k - \eta_k \boldsymbol{W}\nabla f(\boldsymbol{x}^k), \qquad [14]$$

where $\boldsymbol{W} = \boldsymbol{I} - \frac{\boldsymbol{1}\boldsymbol{1}^\top}{N}$.

Based on the dynamics of consensus errors $\boldsymbol{e}^k$ in [14], we can prove that the consensus error $\|\boldsymbol{e}^k\|^2$ will converge almost surely to zero, i.e., all $\boldsymbol{x}_i^k$ will almost surely converge to the same value.

Theorem 1. (Consensus of Optimization Variables) *Let Assumptions 1, 2, and 3 hold. Given any probability $0 < p < 1$, Algorithm 1 with our stepsize strategy (which takes p as input) ensures consensus error less than $\mathcal{O}\left(\frac{1}{k}\right)^{\frac{\alpha}{3}}$ with probability at least $1-p$ for all $k \geqslant t_0$, where t_0 is given in step 1 and step 2 of the stepsize strategy, respectively:*

$$\mathbb{P}\left(\left\|\boldsymbol{e}^k\right\|^2 \leqslant \mathcal{O}\left(\frac{1}{k}\right)^{\frac{\alpha}{3}}, \ \mathit{for\ all}\ k \geqslant t_0\right) \geqslant 1-p. \qquad [15]$$

Moreover, all agents' optimization variables converge to the same value almost surely, i.e., the consensus error $\|\boldsymbol{e}^k\|$ converges almost surely to zero.

Based on the consensus result in Theorem 1, we can further prove that Algorithm 1 ensures all local optimization variables to converge to a first-order stationary point under the given quantization scheme and stepsize strategy:

Theorem 2. (Converging to a First-order Stationary Point) *Let Assumptions 1, 2, and 3 hold. Given any probability* $0 < p < 1$, *Algorithm 1 with our stepsize strategy (which takes p as input) ensures that the gradient* $\|\nabla F\left(\bar{\boldsymbol{x}}^k\right)\|$ *will converge to zero with a probability no less than* $1 - p$, *i.e.,*

$$\mathbb{P}\left(\lim_{k\to\infty} \left\|\nabla F\left(\bar{\boldsymbol{x}}^k\right)\right\|^2 = 0\right) \geqslant 1 - p. \qquad [16]$$

It is worth noting that due to the employment of ε_k (which gradually suppresses the influence of quantization errors) and the unbiasedness of the quantization scheme (the quantization error has a mathematical expectation equal to zero), our algorithm ensures convergence to an exact minimum that has a zero gradient value (with zero steady-state error). In fact, the absence of steady-state error under unbiased quantization has been obtained in the literature such as QSGD (25) and TernGrad (24).

Escaping Saddle Points and Converging to a Second-order Stationary Point

According to Definition 2, saddle points are undesirable states that stall the iteration process. Given that 1) individual agents' local optimization variables $\boldsymbol{x}_i^k$ quickly converge to the same value (reach consensus) according to Theorem 1; and 2) before reaching consensus, inter-agent interaction acts as an additional force (besides the gradient) to keep individual states $\boldsymbol{x}_i^k$ evolving and hence to avoid them from being trapped at any fixed value, we can only consider the saddle-point problem when the states are consensual. In fact, even after all states have reached consensus, since the force brought by inter-agent iteration diminishes at a slower rate than the driven force of the gradient (ε_k decays slower than η_k in our stepsize strategy), the quantized interaction will have enough perturbations on individual agents' optimization variables to efficiently avoid them from being trapped at any saddle point. Formally, we can prove the following results:

Theorem 3 (Escaping Saddle Points). *Let Assumptions 1, 2, and 3 hold. Given any probability* $0 < p < 1$, *Algorithm 1 with our stepsize strategy (which takes p as input) ensures that any "holding stage" in the stepsize strategy reduces the objective function by a substantial amount. More specifically, for any* $i \in \{1, 2, \dots I\}$, *after no more than* $K = \mathcal{O}(\frac{1}{\varepsilon_{t_i}\sqrt{\rho\epsilon}})$ *iterations with the stepsizes held at* $\{\varepsilon_{t_i}, \eta_{t_i}\}$, *Algorithm 1 ensures that with a substantial probability, the objective function has a significant decrease, i.e.,*

$$\mathbb{P}\left(F\left(\bar{\boldsymbol{x}}^{t_i+K}\right) - F\left(\bar{\boldsymbol{x}}^{t_i}\right) \leqslant -Q\right) \geqslant \frac{1}{3} - p, \qquad [17]$$

where Q is a constant satisfying $Q = \mathcal{O}\left(\sqrt{\frac{\epsilon^3}{\rho}}\right)$.

It is worth noting that although the inter-agent interaction (after quantization) can perturb individual agents' optimization variables from staying at any fixed point in the state space, it cannot ensure escaping from a saddle point since the state may evolve in and out of the neighborhood of a saddle point. To facilitate escaping from saddle points, we have to make full use of the existence of descending directions at strict saddle points. More specifically, in our design of the quantization scheme and stepsize strategy, we exploit random quantization to ensure that perturbations exist in every direction and use switching quantization levels to ensure that the amplitude of such perturbations is persistent. To ensure a sufficient integration of the perturbation effect into the iterative dynamics and make it last long enough to evade a saddle point, we hold the stepsizes ε_k and η_k constant for a judiciously calculated period of time (see Fig. 2).

In fact, besides evading a saddle point, Theorem 3 establishes that in each "holding stage" where the stepsizes ε_k and η_k are held constant, the algorithm is guaranteed to decrease in the function value for a significant amount. Therefore, if we can have an estimation of a lower bound on the optimal function value $f^\star$, we can repeat this holding stage multiple times to ensure avoidance of all potentially encountered saddle points, and hence, to ensure convergence to a second-order stationary point.

In practice, during the algorithm's iterations, encountered points can be classified into two categories: points with relatively large gradients $\|\nabla F(\bar{\boldsymbol{x}})\| > \epsilon$ and points with small gradients $\|\nabla F(\bar{\boldsymbol{x}})\| \leqslant \epsilon$, i.e., saddle points. We can prove that within the t_I iterations defined in the stepsize strategy, the algorithm will encounter a second-order stationary point at least once:

Theorem 4. (Converging to a Second-order Stationary Point) *Let Assumptions 1, 2, and 3 hold. For any* $\epsilon > 0$ *and any given probability* $0 < p < 1$, *our stepsize strategy (which takes p as input) ensures that Algorithm 1 will visit an* $\epsilon-$*second-order stationary point at least once with probability at least* $1 - p$ *in* t_I *iterations stated in the stepsize strategy.*

From the derivation of Theorem 4 in the Supporting Information, we can obtain that it takes the following number of iterations to find an $\epsilon-$second-order stationary point:

$$\mathcal{O}\left(\frac{1}{\epsilon^2}\max\left\{(Nd\ell^2)^{\frac{3}{2\alpha}}, (\frac{(Nd\ell^2)^{\frac{2}{3}}}{\epsilon^{2.5}})^{\frac{1}{2\alpha-\beta}}, (\frac{(Nd)^{1/6}}{\epsilon^{1/4}\ell^{2/3}})^{\frac{1}{\beta-4\alpha/3}}\right\}\right), \qquad [18]$$

where N is the number of agents participating in the distributed optimization, d is the dimension of the optimization variable, ℓ is the size of the quantization interval, and α and β are the parameters in stepsizes ε_k and η_k, respectively (note that $2\alpha > \beta > \frac{4}{3}\alpha$ holds according to our stepsize strategy). Therefore, the computational complexity of our algorithm increases polynomially with increases in the network size N and the dimension of optimization variation d. It is worth noting that the computational complexity does not increase monotonically with the size of the quantization interval ℓ: both a too small ℓ and a too large ℓ lead to a high computational complexity. This is understandable since a too small quantization interval ℓ leads to too small quantization errors to stir the evolution of optimization variables, which makes it hard to evade saddle points; whereas a too large quantization interval ℓ results in too much noise injected into the system, which is also detrimental to the convergence of all agents to a stationary point.

Experiments

In this section, we evaluate the performance of the proposed algorithm in five nonconvex-optimization application examples with different scales and complexities. In all five experiments, we consider five agents interacting on the topology depicted in Fig. 3.

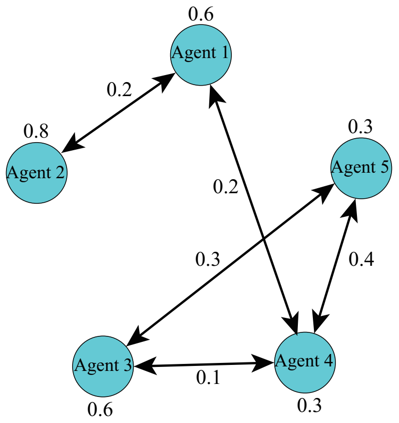

Fig. 3. Interaction weights of five agents

Binary Classification. In this experiment, we consider a simple $\{0, 1\}$ – classification neural network with a single linear hidden layer and a logistic activation function. We use the cross-entropy loss function to train the network (see (49) for details). We denote the feature vector as $\boldsymbol{h} \in \mathbb{R}^M$ and the binary class label as $y \in \{-1, 1\}$. For the fully connected hidden layer, we represent the weights as $\boldsymbol{W}_2 \in \mathbb{R}^{L\times M}$ and $\boldsymbol{W}_1 \in \mathbb{R}^L$. The output is of the form:

$$\hat{y} = \frac{1}{1 + e^{-\langle \boldsymbol{h}, \boldsymbol{W}_2^\top \boldsymbol{W}_1 \rangle}} \qquad [19]$$

Under the commonly used cross-entry loss function, the objective function is of the following form:

$$L\left(\boldsymbol{W}_1, \boldsymbol{W}_2\right) = \log\left(1 + e^{-y\langle \boldsymbol{h}, \boldsymbol{W}_2^\top \boldsymbol{W}_1 \rangle}\right) \qquad [20]$$

To visualize the evolution of optimization variables under our algorithm, we consider the scalar case with $L = M = 1$ and plot the expected loss function (with regulation) in Fig. 4:

$$F(w_1, w_2) = \mathbb{E}\left[L\left(w_1, w_2\right)\right] + \frac{\rho}{2}\left(\parallel w_1 \parallel^2 + \parallel w_2 \parallel^2\right) \qquad [21]$$

When the training samples satisfy $\mathbb{E}\left[y\boldsymbol{h}\right] = 1$ and the regularization parameter is set to $\rho = 0.1$, it becomes apparent that $(w_1, w_2) = (0, 0)$ is a saddle point. We can also verify that this saddle point is a strict saddle point since its Hessian has a negative eigenvalue of -0.4. In our numerical experiment, we purposely initialize all the agents from the strict saddle point $(0, 0)$, and plot in Fig. 4 the evolution of each agent under stepsize parameters $\alpha = 0.62$, $\beta = 0.94$, $c_1 = 0.03$, and $c_2 = 0.3$. It can be seen that due to the quantization effect, all five agents collectively move along the descending direction, implying that our algorithm can effectively evade saddle points.

Matrix Factorization. In this experiment, we consider the 'Matrix Factorization' problem using the 'MovieLen 100K' dataset and compare the performance of the proposed algorithm with a commonly used algorithm in (50). In the matrix factorization problem, given a matrix $\boldsymbol{A} \in \mathbb{R}^{m\times n}$ and $r < \min\{m, n\}$, the goal is to find two matrices $\boldsymbol{U} \in \mathbb{R}^{m\times r}$ and $\boldsymbol{V} \in \mathbb{R}^{n\times r}$ such that $F(\boldsymbol{U}, \boldsymbol{V}) = \frac{\|\boldsymbol{U}\boldsymbol{V}^\top - \boldsymbol{A}\|_F^2}{2}$ is minimized. However, due to the invariance property (51), the matrix factorization problem cannot be considered strongly convex (or even convex) in any local neighborhood around its minima. In our numerical experiments, we implement both our algorithm and the algorithm in (50). In order to ensure a fair comparison, both algorithms share the same set of learning rates ($\alpha = 0.62$, $\beta = 0.94$, $c_1 = 0.3$, $c_2 = 0.3$). For the quantization scheme, we chose ℓ such that all quantized outputs are representable using a binary string of 9 bits. We spread the data evenly across the five agents.

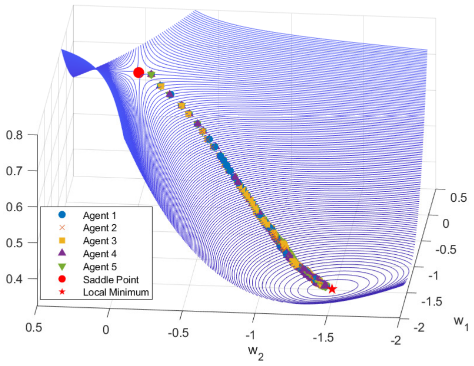

Fig. 4. Trajectories of all five agents when initialized on the saddle point (0,0). Note that all trajectories overlap with each other, implying perfect consensus among the agents.

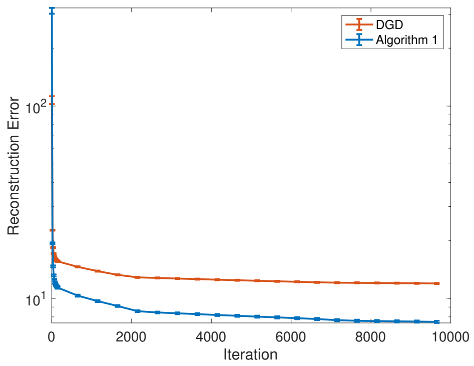

Fig. 5. Comparison of the objective function value between the proposed Algorithm 1 and the existing algorithm DGD in (50).

Fig. 5 shows the evolution of the objective function values under our algorithm and the existing algorithm DGD in (50), respectively. It is clear that our algorithm gives a much smaller cost value. To show that this is indeed due to different convergence properties between our algorithm and DGD, in Fig. 6, we plot the distance between learned parameters and the global optimal parameter, which is obtained using centralized optimization. It is clear that our algorithm indeed converges to a much better solution than DGD, likely due to its ability to evade saddle points.

Convolutional Neural Network. For this experiment, we consider the training of a convolutional neural network (CNN) for the classification of the CIFAR-10 dataset, which contains 50,000 training images across 10 different classes. We evenly spread the CIFAR-10 dataset to the five agents, and set the batch size as 32. Our baseline CNN architecture is a deep network ResNet-18, the training of which is a highly nonconvex problem characterized by the presence of many

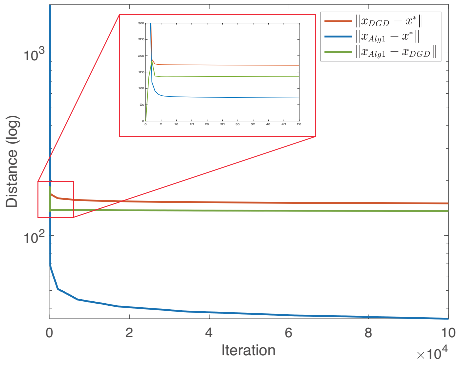

Fig. 6. Comparison of the distance between learned parameters and the actual optimal solution x^* (obtained using centralized optimization). The learned parameters in our algorithm are represented as x_{Alg1}, and the learned parameters in the existing algorithm DGD are represented by x_{DGD}. It can be seen that our algorithm does converge to a better solution than DGD.

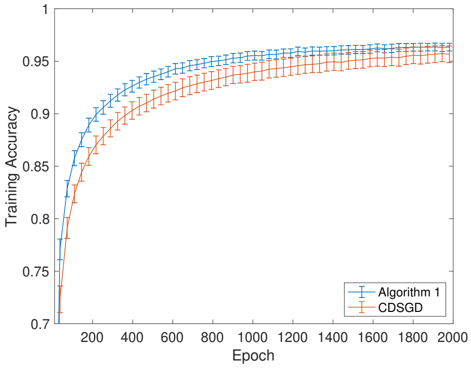

Fig. 7. Comparison of training accuracy between the proposed algorithm and a commonly used algorithm CDSGD from (52).

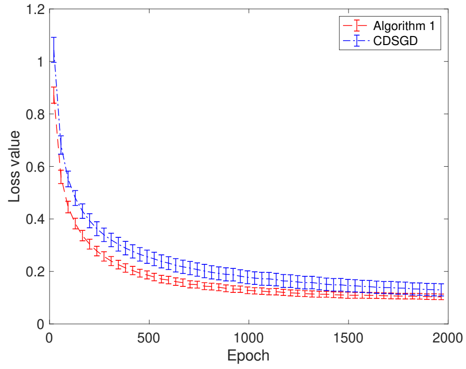

Fig. 8. Comparison of loss function value between the proposed algorithm and a commonly used algorithm CDSGD from (52).

saddle points (16). In the experiments, we train the CNN using both the proposed Algorithm 1 and the decentralized optimization algorithm CDSGD proposed in (52). In order to ensure fairness in comparison, both algorithms use the same set of learning rates ($\alpha = 0.62$, $\beta = 0.94$, $c_1 = 0.5$, $c_2 = 0.3$). The quantization interval ℓ is set such that all quantized outputs are representable using a binary string of 10 bits.

The evolution of the training accuracies and loss-function values averaged over 10 runs are illustrated in Fig. 7 and Fig. 8, respectively. It is evident that Algorithm 1 achieves lower loss function values more rapidly compared to CDSGD. This difference indicates that controlled quantization effects in our algorithm can aid in evading saddle points and discovering better function values.

Tensor Decomposition. In this experiment, we consider Tucker tensor decomposition on the neural dataset in (53). For N neurons over K experimental trials, when each trial has T time samples, the recordings of firing activities can be represented as an $N \times T \times K$ array, which is also called a third-order tensor (54). Each element in this tensor, $x_{n,t,k}$, denotes the firing rate of neuron n at time t within trial k. Tucker tensor decomposition decomposes a tensor into a core tensor multiplied by a matrix along each mode. Following (54), we consider the tensor decomposition problem for a tensor recording $\mathscr{X} \in \mathbb{R}^{50\times 500\times 100}$ of neural firing activities:

$$\mathscr{X} \approx \mathcal{T} \times_1 A \times_2 B \times_3 C = \sum_{n=1}^{N} \sum_{t=1}^{T} \sum_{k=1}^{K} t_{n,k,t} a_n \circ b_t \circ c_k, \quad [22]$$

where $\circ$ represents the vector outer product, $\times_i$ (with $i = \{1, 2, 3\}$) denotes the i-mode matrix product, $\mathcal{T} \in \mathbb{R}^{5\times 5\times 5}$ is the core tensor, and $A \in \mathbb{R}^{50\times 5}$, $B \in \mathbb{R}^{500\times 5}$ and $C \in \mathbb{R}^{100\times 5}$ are the three factors for Tucker decomposition. The goal of tensor decomposition is to minimize the normalized reconstruction error $\mathcal{E} = \left(\| \mathscr{X} - \mathcal{T} \times_1 A \times_2 B \times_3 C \|_F^2\right) / \| \mathscr{X} \|_F^2$, where the subscript F denotes the Frobenius norm. It is well known that the tensor decomposition problem is inherently susceptible to the saddle point issue (15).

We implement both the DGD algorithm in (50) and our Algorithm 1 to solve the tensor decomposition problem. For the DGD algorithm, we use the largest constant stepsize that can still ensure convergence, and for our algorithm, we set the stepsize parameters as $\alpha = 0.61$, $\beta = 0.92$, $c_1 = 0.03$, and $c_2 = 0.3$. The quantization interval ℓ is set such that all quantized outputs are representable using a binary string of 6 bits.

The evolution of the reconstruction error for the two algorithms under 50 runs is shown in Fig. 9. It is clear that our algorithm finds better optimization solutions by effectively evading saddle points.

Robust Principal Component Analysis (PCA). In this experiment, we consider the problem of background subtraction in computer vision using robust PCA. Compared with the conventional PCA, robust PCA can provide a low-dimensional approximation that is more robust to outliers in data samples. For a given sequence of images (video), we employ robust PCA to separate moving objectives in the video from the static background. More specifically, for a given sequence of images represented as a data matrix $\boldsymbol{M} \in \mathbb{R}^{m\times n}$, we

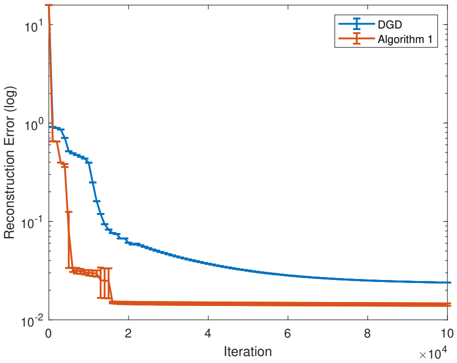

Fig. 9. Comparison of reconstruction error in tensor decomposition between the proposed Algorithm 1 and the existing algorithm DGD in (50).

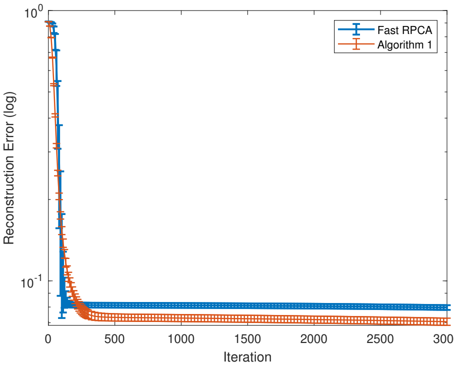

Fig. 10. Comparison of the reconstruction error in Robust PCA between the proposed Algorithm 1 and the existing algorithm Fast RPCA in (57).

use robust PCA to decompose $\boldsymbol{M}$ into a low-rank matrix $\boldsymbol{U}\boldsymbol{V}^\top$ (representing the background) and a sparse matrix $\boldsymbol{S}$ (representing moving objects), where $\boldsymbol{U} \in \mathbb{R}^{m\times r}$, $\boldsymbol{V} \in \mathbb{R}^{n\times r}$, $\boldsymbol{S} \in \mathbb{R}^{m\times n}$ and $r \ll \min\{m,n\}$. Mathematically, the problem can be formulated as the following optimization problem (55):

$$\begin{aligned} &\min_{\boldsymbol{U},\boldsymbol{V}} f(\boldsymbol{U},\boldsymbol{V}) + \mu_2 \|\boldsymbol{U}^\top\boldsymbol{U} - \boldsymbol{V}^\top\boldsymbol{V}\|_F^2, \\ &f(\boldsymbol{U},\boldsymbol{V}) = \min_{\boldsymbol{S}\in\mathcal{S}_{\bar{\alpha}}} \frac{1}{2}\|\boldsymbol{U}\boldsymbol{V}^\top + \boldsymbol{S} - \boldsymbol{M}\|_F^2, \end{aligned} \qquad [23]$$

where μ_2 is a constant and $\mathcal{S}_{\bar{\alpha}}$ represents the set of matrices with at most $\bar{\alpha}-$fraction of nonzero entries in every column and every row.

In our experiment, we use the "WallFlower" datasets from Microsoft (56). We randomly assign 200 image frames with 56 × 56 pixels to each agent, resulting in the data matrix $\boldsymbol{M}_i$ of agent i being of dimensions $m = 9408$ and $n = 200$. We set μ_2 to 0.01, $\bar{\alpha} = 0.2$, and $r = 30$, and then solve [23] using the gradient descent based algorithm ("Fast RPCA") in (57). Fast RPCA employs a sorting-based estimator to generate an initial estimate $\boldsymbol{S}_0$ and then it employs singular value decomposition to generate the corresponding initial values of $\boldsymbol{U}_0$ and $\boldsymbol{V}_0$. Fast RPCA alternates between taking gradient steps for $\boldsymbol{U}$ and $\boldsymbol{V}$, and computing a sparse estimator to adjust $\boldsymbol{S}$. In the experiment, we use the best constant stepsize that we can find for Fast RPCA (the largest stepsize that can still ensure convergence). For our algorithm, we set the stepsize parameters as $\alpha = 0.61$, $\beta = 0.92$, $c_1 = 0.003$, and $c_2 = 0.3$. The quantization interval ℓ is set such that all quantized outputs are representable using a binary string of 5 bits.

Fig. 10 shows the evolutions of the reconstruction error $\mathcal{E} = \sum_{i=1}^N \|\boldsymbol{M}_i - \boldsymbol{U}_i\boldsymbol{V}_i^\top - \boldsymbol{S}_i\|/\|\boldsymbol{M}_i\|_F^2$ under our algorithm and Fast RPCA in (57), respectively. It is clear that our algorithm is capable of identifying superior solutions that yield a smaller reconstruction error. This implies that our algorithm can locate more favorable stationary points by effectively avoiding strict saddle points (it has been proven in (15) that all saddle points in robust PCA are strict saddle points).

Discussions

On Comparison with Other Stepsize Strategies. To test if our stepsize strategy leads to a reduced convergence speed compared with existing counterparts which do not consider saddle-point avoidance, we also conduct experiments using the tensor decomposition problem to compare the convergence speed under our stepsize strategy, the constant stepsize strategy, a random stepsize strategy, and the conventional diminishing stepsize strategy. For the constant stepsize case, we use the largest constant stepsize that does not lead to divergence, and for the random stepsize strategy, we select the stepsize values in the "hold" stages of our approach randomly from the reference functions $\frac{0.03}{1+0.3t^{0.61}}$ and $\frac{0.03}{1+0.3t^{0.92}}$. For the diminishing stepsize case, we use the reference functions as the stepsizes, which are commonly used in distributed optimization. The simulation results in Fig. S1 of the Supporting Information show that our algorithm can provide similar or even faster convergence speeds, and hence show that our approach does not trade convergence speed for saddle-point avoidance.

On Comparison with the Log-scale Quantization. It is worth noting that recently (58) and (59) propose to use log-scale quantization in distributed optimization and prove that accurate convergence can be ensured when the objective functions are convex. However, the log-scale quantization scheme is not appropriate for the saddle-point avoidance problem in distributed nonconvex optimization. This is because to enable saddle-point avoidance, we have to keep the magnitude of quantization error large enough to perturb the optimization variable, no matter what the value of the optimization variable is (because we do not know where the saddle-point is). In fact, this is why we introduce the periodic switching between two sets of quantization levels in our quantization scheme (to avoid the possibility that a quantization input coincides with an endpoint of a quantization interval and results in a zero quantization error). However, the log-scale quantization scheme results in a quantization error that can be arbitrarily small when the quantization input is arbitrarily close to zero, meaning that the quantization-induced perturbation becomes negligible when the quantization input is close to zero, making it inappropriate for saddle-point avoidance. In fact, our experimental results using the binary classification problem in Fig. S2 of the Supporting Information also confirm that the log-scale quantization scheme cannot provide comparable performance with our proposed quantization scheme.

On Applicability to High-Order Optimization Methods. Given that additive noises have been proven effective in evading saddle points in second-order optimization algorithms as well (see, e.g., (60)), our quantization effect based approach is well positioned to help saddle-point avoidance in second-order nonconvex optimization algorithms. To confirm this point, we apply the quantization scheme to second-order Newton-method based distributed optimization for the binary classification problem (see details in the section "Experimental Results Based on the Newton Method" on page 19 of the Supporting Information). The results in Fig. S3 in the Supporting Information confirm that our quantization scheme does significantly enhance the quality of the solution by evading saddle points compared with the case without quantization effects. We plan to systematically investigate exploiting quantization effects in high-order optimization algorithms to evade saddle points in future work.

On Relaxing the Smoothness Assumption. In the theoretical analysis, we assume that the objective functions are Lipschitz continuous. Given that "generalized gradients" (61) have been proven effective to address non-smooth objective functions in convex optimization, it is tempting to investigate if the generalized gradient approach can be exploited to address nonconvex and non-smooth objective functions. Unfortunately, (62) proves that in general nonconvex and non-smooth optimization, for any $\epsilon \in [0, 1)$, there is a more than 50% probability that an ϵ-first-order-stationary point (defined in the sense of the generalized gradient, usually called Clarke stationary point) can never be found by any finite-time algorithm. In future work, we plan to explore if some subclasses of nonconvex and non-smooth objective functions can be addressed using the generalized gradient approach.

Conclusions

Saddle-point avoidance is a fundamental problem in nonconvex optimization. Compared with the centralized optimization case, saddle-point avoidance in distributed optimization faces unique challenges due to the fact that individual agents can only access local gradients, which may be significantly different from the global gradient (which actually carries information about saddle points). We show that quantization effects, which are unavoidable in any digital communications, can be exploited without additional cost to evade saddle points in distributed nonconvex optimization. More specifically, by judiciously co-designing the quantization scheme and the stepsize strategy, we propose an algorithm that can ensure saddle-point avoidance and convergence to second-order stationary points in distributed nonconvex optimization. Given the widespread applications of distributed nonconvex optimization in numerous engineered systems and deep learning, the results are expected to have broad ramifications in various fields involving nonconvex optimization. Numerical experimental results using distributed optimization and learning applications on benchmark datasets confirm the effectiveness of the proposed algorithm.

ACKNOWLEDGMENTS. The work was supported in part by the National Science Foundation under Grants CCF-2106293, CCF-2215088, and CNS-2219487.

1. D Bertsekas, J Tsitsiklis, *Parallel and distributed computation: numerical methods.* (Athena Scientific), (2015).
2. A Nedić, A Ozdaglar, Distributed subgradient methods for multi-agent optimization. *IEEE Transactions on Autom. Control.* **54**, 48–61 (2009).
3. A Nedić, A Ozdaglar, PA Parrilo, Constrained consensus and optimization in multi-agent networks. *IEEE Transactions on Autom. Control.* **55**, 922–938 (2010).
4. K Srivastava, A Nedić, Distributed asynchronous constrained stochastic optimization. *IEEE J. Sel. Top. Signal Process.* **5**, 772–790 (2011).
5. W Shi, Q Ling, G Wu, W Yin, Extra: An exact first-order algorithm for decentralized consensus optimization. *SIAM J. on Optim.* **25**, 944–966 (2015).
6. J Xu, S Zhu, YC Soh, L Xie, Convergence of asynchronous distributed gradient methods over stochastic networks. *IEEE Transactions on Autom. Control.* **63**, 434–448 (2017).
7. G Qu, N Li, Harnessing smoothness to accelerate distributed optimization. *IEEE Transactions on Control. Netw. Syst.* **5**, 1245–1260 (2017).
8. R Xin, UA Khan, A linear algorithm for optimization over directed graphs with geometric convergence. *IEEE Control. Syst. Lett.* **2**, 315–320 (2018).
9. W Shi, Q Ling, K Yuan, G Wu, W Yin, On the linear convergence of the ADMM in decentralized consensus optimization. *IEEE Transactions on Signal Process.* **62**, 1750–1761 (2014).
10. C Zhang, M Ahmad, Y Wang, ADMM based privacy-preserving decentralized optimization. *IEEE Transactions on Inf. Forensics Secur.* **14**, 565–580 (2018).
11. G Tychogiorgos, A Gkelias, KK Leung, A non-convex distributed optimization framework and its application to wireless ad-hoc networks. *IEEE Transactions on Wirel. Commun.* **12**, 4286–4296 (2013).
12. KI Tsianos, S Lawlor, MG Rabbat, Consensus-based distributed optimization: Practical issues and applications in large-scale machine learning in *2012 50th Annual Allerton Conference on Communication, Control, and Computing.* (IEEE), pp. 1543–1550 (2012).
13. M Fazel, R Ge, S Kakade, M Mesbahi, Global convergence of policy gradient methods for the linear quadratic regulator in *International Conference on Machine Learning.* (PMLR), pp. 1467–1476 (2018).
14. K Zhang, B Hu, T Başar, Policy optimization for $\mathcal{H}$_2 linear control with $\mathcal{H}$_∞ robustness guarantee: Implicit regularization and global convergence. *SIAM J. on Control. Optim.* **59**, 4081–4109 (2021).
15. R Ge, F Huang, C Jin, Y Yuan, Escaping from saddle points—online stochastic gradient for tensor decomposition in *Conference on Learning Theory.* (PMLR), pp. 797–842 (2015).
16. YN Dauphin, et al., Identifying and attacking the saddle point problem in high-dimensional non-convex optimization. *Adv. Neural Inf. Process. Syst.* **27** (2014).
17. A Choromanska, M Henaff, M Mathieu, GB Arous, Y LeCun, The loss surfaces of multilayer networks in *Artificial Intelligence and Statistics.* (PMLR), pp. 192–204 (2015).
18. Y Nesterov, BT Polyak, Cubic regularization of Newton method and its global performance. *Math. Program.* **108**, 177–205 (2006).
19. FE Curtis, DP Robinson, M Samadi, A trust region algorithm with a worst-case iteration complexity of $\mathcal{O}(\epsilon^{-3/2})$ for nonconvex optimization. *Math. Program.* **162**, 1–32 (2017).
20. Z Tang, S Shi, X Chu, W Wang, B Li, Communication-efficient distributed deep learning: A comprehensive survey. *arXiv preprint arXiv:2003.06307* (2020).
21. SS Du, et al., Gradient descent can take exponential time to escape saddle points. *Adv. Neural Inf. Process. Syst.* **30** (2017).
22. JD Lee, M Simchowitz, MI Jordan, B Recht, Gradient descent only converges to minimizers in *Conference on Learning Theory.* (PMLR), pp. 1246–1257 (2016).
23. A Daneshmand, G Scutari, V Kungurtsev, Second-order guarantees of distributed gradient algorithms. *SIAM J. on Optim.* **30**, 3029–3068 (2020).
24. W Wen, et al., Terngrad: Ternary gradients to reduce communication in distributed deep learning. *Adv. Neural Inf. Process. Syst.* **30** (2017).
25. D Alistarh, D Grubic, J Li, R Tomioka, M Vojnovic, QSGD: Communication-efficient SGD via gradient quantization and encoding. *Adv. Neural Inf. Process. Syst.* **30** (2017).
26. A Kashyap, T Başar, R Srikant, Quantized consensus. *2006 IEEE Int. Symp. on Inf. Theory* pp. 635–639 (2006).
27. MG Rabbat, RD Nowak, Quantized incremental algorithms for distributed optimization. *IEEE J. on Sel. Areas Commun.* **23**, 798–808 (2005).
28. M El Chamie, J Liu, T Başar, Design and analysis of distributed averaging with quantized communication. *IEEE Transactions on Autom. Control.* **61**, 3870–3884 (2016).
29. J Wang, N Elia, A control perspective for centralized and distributed convex optimization in *2011 50th IEEE Conference on Decision and Control and European Control Conference.* (IEEE), pp. 3800–3805 (2011).
30. M Zhu, S Martínez, On distributed convex optimization under inequality and equality constraints. *IEEE Transactions on Autom. Control.* **57**, 151–164 (2011).
31. SS Kia, J Cortés, S Martínez, Distributed convex optimization via continuous-time coordination algorithms with discrete-time communication. *Automatica* **55**, 254–264 (2015).
32. L Su, NH Vaidya, Fault-tolerant multi-agent optimization: optimal iterative distributed algorithms in *Proceedings of the 2016 ACM Symposium on Principles of Distributed Computing.* pp. 425–434 (2016).
33. Q Jia, W Chen, Y Zhang, H Li, Fault reconstruction and fault-tolerant control via learning observers in takagi–sugeno fuzzy descriptor systems with time delays. *IEEE Transactions on Ind. Electron.* **62**, 3885–3895 (2015).
34. Y Wang, B Jiang, ZG Wu, S Xie, Y Peng, Adaptive sliding mode fault-tolerant fuzzy tracking control with application to unmanned marine vehicles. *IEEE Transactions on Syst. Man, Cybern. Syst.* **51**, 6691–6700 (2020).
35. A Koloskova, SU Stich, M Jaggi, Decentralized stochastic optimization and gossip algorithms with compressed communication in *International Conference on Machine Learning.* (PMLR), Vol. 97, pp. 3479–3487 (2019).

36. K Yuan, et al., Revisiting optimal convergence rate for smooth and non-convex stochastic decentralized optimization. *Adv. Neural Inf. Process. Syst.* **35**, 36382–36395 (2022).
37. J Zeng, W Yin, On nonconvex decentralized gradient descent. *IEEE Transactions on Signal Process.* **66**, 2834–2848 (2018).
38. Y Nesterov, Squared functional systems and optimization problems in *High Performance Optimization.* (Springer), pp. 405–440 (2000).
39. J Sun, Q Qu, J Wright, Complete dictionary recovery over the sphere I: Overview and the geometric picture. *IEEE Transactions on Inf. Theory* **63**, 853–884 (2017).
40. N Boumal, V Voroninski, A Bandeira, The non-convex burer-monteiro approach works on smooth semidefinite programs. *Adv. Neural Inf. Process. Syst.* **29** (2016).
41. R Ge, C Jin, Y Zheng, No spurious local minima in nonconvex low rank problems: A unified geometric analysis in *International Conference on Machine Learning.* (PMLR), pp. 1233–1242 (2017).
42. S Vlaski, AH Sayed, Distributed learning in non-convex environments—part II: Polynomial escape from saddle-points. *IEEE Transactions on Signal Process.* **69**, 1257–1270 (2021).
43. A Reisizadeh, A Mokhtari, H Hassani, R Pedarsani, An exact quantized decentralized gradient descent algorithm. *IEEE Transactions on Signal Process.* **67**, 4934–4947 (2019).
44. Y Wang, A Nedić, Tailoring gradient methods for differentially-private distributed optimization. *IEEE Transactions on Autom. Control.* (2023).
45. Y Wang, Ensure differential privacy and convergence accuracy in consensus tracking and aggregative games with coupling constraints. *arXiv preprint arXiv:2210.16395* (2022).
46. Y Wang, HV Poor, Decentralized stochastic optimization with inherent privacy protection. *IEEE Transactions on Autom. Control.* **68**, 2293–2308 (2022).
47. TT Doan, ST Maguluri, J Romberg, Convergence rates of distributed gradient methods under random quantization: A stochastic approximation approach. *IEEE Transactions on Autom. Control.* **66**, 4469–4484 (2020).
48. C Jin, R Ge, P Netrapalli, SM Kakade, MI Jordan, How to escape saddle points efficiently in *International Conference on Machine Learning.* (PMLR), pp. 1724–1732 (2017).
49. S Vlaski, AH Sayed, Distributed learning in non-convex environments—part I: Agreement at a linear rate. *IEEE Transactions on Signal Process.* **69**, 1242–1256 (2021).
50. K Yuan, Q Ling, W Yin, On the convergence of decentralized gradient descent. *SIAM J. on Optim.* **26**, 1835–1854 (2016).
51. Z Zhu, Q Li, G Tang, MB Wakin, The global optimization geometry of low-rank matrix optimization. *IEEE Transactions on Inf. Theory* **67**, 1308–1331 (2021).
52. Z Jiang, A Balu, C Hegde, S Sarkar, Collaborative deep learning in fixed topology networks. *Adv. Neural Inf. Process. Syst.* **30** (2017).
53. M Bashiri, A short tutorial on implementing canonical polyadic (cp) tensor decomposition in python (https://github.com/mohammadbashiri/tensor-decomposition-in-python) (2019).
54. AH Williams, et al., Unsupervised discovery of demixed, low-dimensional neural dynamics across multiple timescales through tensor component analysis. *Neuron* **98**, 1099–1115 (2018).
55. S Ma, NS Aybat, Efficient optimization algorithms for robust principal component analysis and its variants. *Proc. IEEE* **106**, 1411–1426 (2018).
56. Test images for wallflower paper (https://www.microsoft.com/en-us/download/details.aspx?id=54651) (2017).
57. X Yi, D Park, Y Chen, C Caramanis, Fast algorithms for robust PCA via gradient descent. *Adv. Neural Inf. Process. Syst.* **29** (2016).
58. M Doostmohammadian, et al., Distributed anytime-feasible resource allocation subject to heterogeneous time-varying delays. *IEEE Open J. Control. Syst.* **1**, 255–267 (2022).
59. M Doostmohammadian, et al., Fast-convergent anytime-feasible dynamics for distributed allocation of resources over switching sparse networks with quantized communication links in *2022 European Control Conference (ECC).* (IEEE), pp. 84–89 (2022).
60. S Paternain, A Mokhtari, A Ribeiro, A Newton-based method for nonconvex optimization with fast evasion of saddle points. *SIAM J. on Optim.* **29**, 343–368 (2019).
61. J Cortes, Discontinuous dynamical systems. *IEEE Control. Syst. Mag.* **28**, 36–73 (2008).
62. J Zhang, H Lin, S Jegelka, S Sra, A Jadbabaie, Complexity of finding stationary points of nonconvex nonsmooth functions in *International Conference on Machine Learning.* (PMLR), pp. 11173–11182 (2020).