2018 IEEE Conference on Decision and Control (CDC)
Miami Beach, FL, USA, Dec. 17-19, 2018

SUCAG: Stochastic Unbiased Curvature-aided Gradient Method for Distributed Optimization

Hoi-To Wai*, Nikolaos M. Freris†, Angelia Nedić* and Anna Scaglione*
*School of Electrical, Computer and Energy Engineering, Arizona State University, Tempe, AZ 85281, USA.
†Division of Engineering, New York University Abu Dhabi & NYU Tandon School of Engineering
Emails: {htwai,angelia.nedich,Anna.Scaglione}@asu.edu, nf47@nyu.edu

***Abstract*—We propose and analyze a new stochastic gradient method, which we call *Stochastic Unbiased Curvature-aided Gradient* (SUCAG), for finite sum optimization problems. SUCAG constitutes an *unbiased* total gradient tracking technique that uses Hessian information to accelerate convergence. We analyze our method under the general asynchronous model of computation, in which each function is selected infinitely often with possibly unbounded (but sublinear) delay. For strongly convex problems, we establish linear convergence for the SUCAG method. When the initialization point is sufficiently close to the optimal solution, the established convergence rate is only dependent on the condition number of the problem, making it strictly faster than the known rate for the SAGA method.**

Furthermore, we describe a Markov-driven approach of implementing the SUCAG method in a distributed asynchronous multi-agent setting, via gossiping along a random walk on an undirected communication graph. We show that our analysis applies as long as the graph is connected and, notably, establishes an asymptotic linear convergence rate that *is robust to the graph topology*. Numerical results demonstrate the merits of our algorithm over existing methods.

***Index Terms*— Distributed optimization, Incremental methods, Asynchronous algorithms, Randomized algorithms, Multi-agent systems, Machine learning.**

I. Introduction

We consider the finite sum optimization problem:

$$\min_{\boldsymbol{\theta} \in \mathbb{R}^d} \; F(\boldsymbol{\theta}) \quad \text{where} \quad F(\boldsymbol{\theta}) := \frac{1}{N} \sum_{i=1}^{N} f_i(\boldsymbol{\theta}) \; . \tag{1}$$

Each component function $f_i : \mathbb{R}^d \rightarrow \mathbb{R}$ is assumed to be convex and twice continuously differentiable with Lipschitz continuous gradients, while the sum function $F(\boldsymbol{\theta})$ is assumed to be strongly convex. Such problem customarily arises in several multi-agent systems applications [1], e.g., communication networks [2], sensor networks [3], and distributed learning [4]. In a distributed setting, $f_i(\cdot)$ is the *private* function known only to agent i; for example, in a learning task such as empirical risk minimization (ERM) [5], this corresponds to a subset of data gathered by agent i.

This paper develops an efficient distributed algorithm for (1) under a general communication topology. In particular, we adopt a stochastic asynchronous setting that captures activation of agents in networked decision and control applications. To this end, a large body of prior work has focused on developing distributed algorithms based on the average consensus protocol [6], e.g., as introduced by Nedić *et al.* [7]. However, the convergence rate for this class of methods is limited by the diameter of the communication graph [8]. To develop an algorithm that is robust to the graph structure, we consider here an alternative pathway that is akin to that described in [9], [10], for utilizing stochastic algorithms for finite sum minimization via pairwise exchange of information on a graph [1], [11]. Inspired by the classical work of Robbins and Monro [12], stochastic algorithms have been actively pursued for machine learning operations over the last decade, with notable recent examples enlisting stochastic average gradient (SAG) [13] and its unbiased variant (SAGA) [14], and stochastic variance reduction gradient (SVRG) [15]. All these methods (including the one we propose here) are *incremental* [16] and operate using the following key steps, at each iteration: a) a *random* component function is selected, and its gradient is evaluated at the current iterate, b) the obtained gradient value is aggregated into the memory, and c) a new iterate is obtained using the aggregated gradient. Additionally, they require storing the gradient values to allow for an asynchronous implementation.

Inspired by [9], [10], we make an observation that the *random selection-aggregate-update* paradigm can be realized in a fully distributed fashion via a random walk on a graph, where a selected agent forwards information to one of its neighbors, *i.e.,* essentially implementing a Markov-driven token passing protocol; cf. Section II-A. Most notably, while stochastic algorithms such as SAG and SAGA are known to converge linearly for strongly convex problems [13], [14], their rates depend on the number of agents N. For example, SAGA requires $\mathcal{O}((N+\kappa(F))\log(1/\epsilon))$ steps in expectation to reach an ϵ-optimal solution, where $\kappa(F)$ denotes the condition number for the sum function $F(\cdot)$.

In our recent work, we proposed the *Curvature-aided Incremental Aggregated Gradient* (CIAG) [17] method, which utilizes curvature (*i.e.,* Hessian) information to acquire accelerated convergence. Under the asynchronous model of computation [18] (where each component function is selected by the algorithm infinitely often, with a uniformly bounded delay), CIAG was shown to converge linearly at a rate that is strictly faster than SAGA; see also [19] for a similar technique to variance reduction in stochastic gradient methods. Nevertheless, CIAG is not directly amenable to a distributed implementation, where it would require message-passing among agents along a closed spanning walk following a *fixed* sequence. This is typically restrictive, especially in distributed wireless sensor networks where time synchronization is challenging [20] and it is desirable that agents are activated at random time instants for the sake of battery lifetimes.

This paper adopts a stochastic setting where the component functions are selected at random, *i.e.,* it allows for a Markov-driven distributed implementation on a general topology. Moreover, the proposed *Stochastic Unbiased Curvature-aided Gradient* (SUCAG) method introduces an *unbiased* curvature-aided gradient estimator. We analyze the convergence of SUCAG for strongly convex problems and show that it converges linearly *with high probability* (w.h.p.), for a sufficiently small stepsize selection. Besides, for an initialization sufficiently close to optimality, the algorithm requires $\mathcal{O}(\kappa(F)\log(1/\epsilon))$ steps to reach an ϵ-optimal solution, *robust to the graph structure or network size*. This can be beneficial in the prelude of large-scale cyberphysical systems [21] that feature millions of agents, for example in real-time learning from streaming data [22], [23].

Notation. We denote vectors and matrices using boldface lower-case letters and upper-case letters, respectively. We use the notation $(x)_+ := \max\{0, x\}$ for the positive part. We use the standard Bachmann-Landau notation: for non-negative functions f, g, h we write $f(t) = \mathcal{O}(g(t))$ (*resp.* $f(t) = \Omega(h(t))$) when there exists a positive constant c (*resp.* C) such that $f(t) \leq cg(t)$ ($f(t) \geq Ch(t)$). We use $\|\cdot\|$ for the standard Euclidean norm.

A. Overview of Gradient Tracking Techniques

Before we introduce the proposed SUCAG algorithm, we briefly review the curvature-aided gradient tracking technique. In particular, we cast it as a specific variance reduction method within the stochastic gradient paradigm.

When the number of components N is large, and especially in a distributed setting, the total gradient ∇F is costly to evaluate, whence a stochastic gradient (SG) scheme is usually sought to tackle (1) in a tractable manner. To this end, a SG scheme can be described as follows: at iteration $k \in \mathbb{N}$, a *stochastic gradient surrogate* $\boldsymbol{g}^k$ is obtained, satisfying $\mathbb{E}[\boldsymbol{g}^k] = \nabla F(\boldsymbol{\theta}^k)$, $\mathrm{Var}(\boldsymbol{g}^k) \leq \sigma^2$, *i.e.,* an unbiased estimator for $\nabla F(\boldsymbol{\theta}^k)$ with finite error variance. A common implementation selects one of the component functions uniformly at random, *i.e.,* the i_k-th function (where $i_k \in [N]$) at iteration k, and evaluates the gradient $\nabla f_{i_k}(\boldsymbol{\theta}^k)$.

The SG method [12] then solves (1) using the recursion $\boldsymbol{\theta}^{k+1} = \boldsymbol{\theta}^k - \gamma_k \boldsymbol{g}^k_{\mathsf{SG}}$ with $\boldsymbol{g}^k_{\mathsf{SG}} := \nabla f_{i_k}(\boldsymbol{\theta}^k)$, where $\gamma_k > 0$ is the stepsize at the $k-$th iteration: when $\{\gamma_k\}$ satisfies $\sum_k \gamma_k = \infty, \sum_k \gamma_k^2 < \infty$, the SG method converges to an optimal solution of (1) almost surely (a.s.) [12]. However, albeit its simplicity, a major drawback of SG remains that the (expected) convergence rate is *sublinear*, *i.e.,* $\mathbb{E}[\|\boldsymbol{\theta}^\star - \boldsymbol{\theta}^k\|] = \mathcal{O}(1/k)$ (where $\boldsymbol{\theta}^\star$ denotes the optimal solution to (1)) when the objective function is strongly convex [24].

On the other hand, one may adopt an *incremental* approach to the problem. In this purview, a reasonable remedy to the high cost of evaluating the total gradient is to *aggregate* previous gradient component evaluations, *i.e.,* approximate $\nabla f_j(\boldsymbol{\theta}^k) \approx \nabla f_j(\boldsymbol{\theta}^{\tau_j^k})$ where τ_j^k denotes the iteration count when the $j-$th gradient was last evaluated (see (2) for the precise definition of τ_i^k). This is precisely the mechanism of SAGA [14] that selects:

$$\boldsymbol{g}^k_{\mathsf{SAGA}} := \left[\nabla f_{i_k}(\boldsymbol{\theta}^k) - \nabla f_{i_k}(\boldsymbol{\theta}^{\tau_{i_k}^{k-1}})\right] + \frac{1}{N}\sum_{j=1}^N \nabla f_j(\boldsymbol{\theta}^{\tau_j^{k-1}}).$$

Besides, SAG is different only in that it multiplies the first term above by $\frac{1}{N}$ and is, therefore, biased. When each component function f_j is smooth, the approximation error, *i.e.,* the *variance* of the total gradient estimator, is bounded by $\mathcal{O}(\sum_{i=1}^N \|\boldsymbol{\theta}^{\tau_i^k} - \boldsymbol{\theta}^\star\|)$. Thanks to the *variance reduction* property, the SAGA/SAG methods are shown to converge at a linear rate [13], [14] and are therefore faster than SG, at the cost of storing the past gradient component values. It is worthwhile observing that in a multi-agent setting, past values can be stored locally, and aggregated in a recursive manner via message passing. The crucial implementation detail in our algorithm is that aggregation can be performed via simple *gossiping*, *i.e.,* message exchange between an agent and only one of its neighbors.

Under an additional smoothness condition (of Lipschitz-continuous component Hessians), our recent work [17] considered a first-order Taylor approximation to the total gradient in order to further reduce the estimation variance:

$$\boldsymbol{g}^k_{\mathsf{CIAG}} := \frac{1}{N}\sum_{j=1}^N \left[\nabla f_j(\boldsymbol{\theta}^{\tau_j^k}) + \nabla^2 f_j(\boldsymbol{\theta}^{\tau_j^k})(\boldsymbol{\theta}^k - \boldsymbol{\theta}^{\tau_j^k})\right],$$

A distinctive attribute is that the approximation error for $\boldsymbol{g}^k_{\mathsf{CIAG}}$ to the total gradient above can be bounded by $\mathcal{O}(\sum_{j=1}^N \|\boldsymbol{\theta}^k - \boldsymbol{\theta}^{\tau_j^k}\|^2)$ in light of [25, Lemma 1.2.4], *i.e.,* the bound features *squared norm* of the distance to optimality; specifically, when $\|\boldsymbol{\theta}^k - \boldsymbol{\theta}^{\tau_j^k}\|^2 < 1$ (as is the case for large enough k), this bound allows a tighter convergence analysis as compared to SAG/SAGA. In brief, it was shown in [17] that CIAG method exhibits a faster linear convergence rate than SAG *by a factor of* N, under the restriction that the delays in gradient/Hessian evaluation are bounded (*i.e.,* $|\tau_i^k - k| \leq K < \infty$, e.g., when i_k is chosen as $i_k = (k \bmod N) + 1$). Note, however, that in a stochastic setup CIAG is biased, which motivates the development of an unbiased variant in this paper.

II. The SUCAG method

We first introduce SUCAG method as a stochastic gradient algorithm for finite sum optimization (1), and then proceed to present protocols for implementing it in a distributed setting.

We adhere to the stochastic optimization setting: at iteration k, the algorithm selects the i_k-th component function $f_{i_k}(\cdot)$, where $i_k \in \{1, ..., N\}$, and evaluates its gradient/Hessian at the current iterate. At first, we assume that the selection mechanism is independent identically distributed (i.i.d.), where a random integer is chosen *uniformly at random*, *i.e.,* $P(i_k = j) = 1/N$ for all $j = 1, ..., N$. We define the random variable τ_j^k as the iteration number where the $j-$th function was last accessed (including iteration k itself):

$$\tau_j^k := \max\{\ell \geq 0 \;:\; i_\ell = j,\; \ell \leq k\}\,. \tag{2}$$

Algorithm 1 SUCAG method

```
1: Input: Initial point $\boldsymbol{\theta}^0 \in \mathbb{R}^d$
2: Set counter variables $\tau_i^0 := -1$ for all $i$.
3: Set $\nabla f_i(\boldsymbol{\theta}^{-1}) = \mathbf{0}$ and $\nabla^2 f_i(\boldsymbol{\theta}^{-1}) = \mathbf{0}$ for all $i$.
4: for $k = 0, 1, 2, \ldots, K$ do
5:     Choose $i_k \in \{1, \ldots, N\}$ uniformly at random.
6:     Compute $\boldsymbol{g}^k_{\mathsf{SUCAG}}$ using (4), (5).
7:     Update:
           $\boldsymbol{\theta}^{k+1} = \boldsymbol{\theta}^k - \gamma \boldsymbol{g}^k_{\mathsf{SUCAG}}$ .
8:     Update counter variables: $\tau^k_{i_k} \leftarrow k$, and $\tau^k_j \leftarrow \tau^{k-1}_j$
       for all $j \neq i_k$.
9: end for
10: Return $\boldsymbol{\theta}^{K+1}$.
```

Note that $\tau^k_{i_k} = k$ and $\tau^k_j = \tau^{k-1}_j$ for all $j \neq i_k$.

Inspired by the curvature-aided gradient tracking technique [17], the SUCAG method adopts the update rule:

$$\boldsymbol{\theta}^{k+1} = \boldsymbol{\theta}^k - \gamma \boldsymbol{g}^k_{\mathsf{SUCAG}}, \ k \geq 0 \ , \tag{3}$$

where $\gamma > 0$ is a fixed step size and $\boldsymbol{g}^k_{\mathsf{SUCAG}}$ is a curvature-aided approximation on the actual gradient $\nabla F(\boldsymbol{\theta}^k)$. The approximation is given as[1]:

$$\boldsymbol{g}^k_{\mathsf{SUCAG}} := \left[\nabla f_{i_k}(\boldsymbol{\theta}^k) - \boldsymbol{g}^k_{i_k}(\boldsymbol{\theta}^k)\right] + \tfrac{1}{N}\textstyle\sum_{i=1}^N \boldsymbol{g}^k_i(\boldsymbol{\theta}^k) \ , \tag{4}$$

where

$$\boldsymbol{g}^k_i(\boldsymbol{\theta}) := \nabla f_i(\boldsymbol{\theta}^{\tau_i^{k-1}}) + \nabla^2 f_i(\boldsymbol{\theta}^{\tau_i^{k-1}})\big(\boldsymbol{\theta} - \boldsymbol{\theta}^{\tau_i^{k-1}}\big) \ . \tag{5}$$

A key property of $\boldsymbol{g}^k_{\mathsf{SUCAG}}$ is that it is an *unbiased* estimator for $\nabla F(\boldsymbol{\theta}^k)$. To see this, define the filtration $\mathcal{F}_k$ as the collection of random variables (the chosen indices $\{i_k\}$) realized prior to the update of $\boldsymbol{\theta}^{k+1}$, *i.e.,* $\mathcal{F}_k := \sigma(\{i_\ell : \ell = 0, 1, \ldots, k-1\})$; note that this does not include the newly selected index i_k. The conditional expectation is given by:

$$\begin{aligned}\mathbb{E}_k[\boldsymbol{g}^k_{\mathsf{SUCAG}}] &= \tfrac{1}{N}\textstyle\sum_{i=1}^N \left(\boldsymbol{g}^k_i(\boldsymbol{\theta}^k) + \nabla f_i(\boldsymbol{\theta}^k) - \boldsymbol{g}^k_i(\boldsymbol{\theta}^k)\right) \\ &= \nabla F(\boldsymbol{\theta}^k) \ ,\end{aligned} \tag{6}$$

where $\mathbb{E}_k[\cdot] := \mathbb{E}[\cdot|\mathcal{F}_k]$, and therefore SUCAG is unbiased. Note that SUCAG is a natural extension of SAGA to incorporate curvature information.

While our main focus is on applying SUCAG in a distributed setting, we note that one can also implement the SUCAG method as a centralized incremental method just as in CIAG [17]. In this case, the system stores $\mathcal{O}(Nd)$ real numbers in the memory for the previous iterates and the per-iteration complexity is $\mathcal{O}(d^2)$, cf. [17].

A. Distributed Implementation

Consider a *connected* undirected graph $G = (V, E)$ where $V = \{1, \ldots, N\}$ is the set of N agents and $E \subseteq V \times V$ is the edge set that prescribes the agent-to-agent communication pairs. The neighborhood set of i is $\mathcal{N}_i := \{j : (i, j) \in E\}$.

[1]Note that CIAG pre-multiplies the first term with $\frac{1}{N}$, in the same way that SAG does for SAGA.

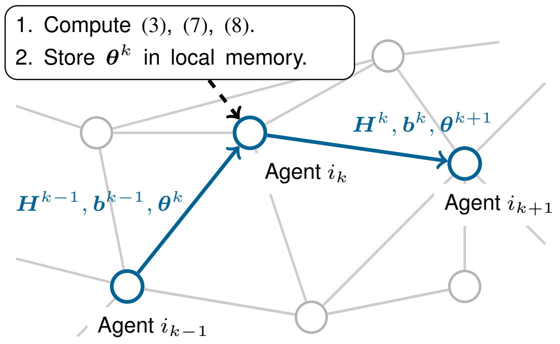
1. Compute (3), (7), (8).
2. Store $\boldsymbol{\theta}^k$ in local memory.
$\boldsymbol{H}^k, \boldsymbol{b}^k, \boldsymbol{\theta}^{k+1}$
$\boldsymbol{H}^{k-1}, \boldsymbol{b}^{k-1}, \boldsymbol{\theta}^k$
Agent i_k
Agent i_{k+1}
Agent i_{k-1}

Fig. 1: Distributed implementation of SUCAG; the agent sequence $\{i_k\}_{k \geq 1}$ corresponds to a random walk on the graph G.

It is easy to verify that:

$$\boldsymbol{g}^k_{\mathsf{SUCAG}} = \left[\nabla f_{i_k}(\boldsymbol{\theta}^k) - \boldsymbol{g}^k_{i_k}(\boldsymbol{\theta}^k)\right] + \boldsymbol{b}^{k-1} + \boldsymbol{H}^{k-1}\boldsymbol{\theta}^k, \tag{7}$$

where

$$\begin{aligned}\boldsymbol{b}^{k-1} &:= \tfrac{1}{N}\textstyle\sum_{i=1}^N \left(\nabla f_i(\boldsymbol{\theta}^{\tau_i^{k-1}}) - \nabla^2 f_i(\boldsymbol{\theta}^{\tau_i^{k-1}})\boldsymbol{\theta}^{\tau_i^{k-1}}\right) \ , \\ \boldsymbol{H}^{k-1} &:= \tfrac{1}{N}\textstyle\sum_{i=1}^N \nabla^2 f_i(\boldsymbol{\theta}^{\tau_i^{k-1}}) \ .\end{aligned}$$

These variables can be computed recursively as follows:

$$\begin{aligned}\boldsymbol{b}^k &:= \boldsymbol{b}^{k-1} + \tfrac{1}{N}\left(\nabla f_{i_k}(\boldsymbol{\theta}^k) - \nabla f_{i_k}(\boldsymbol{\theta}^{\tau_{i_k}^{k-1}})\right) \\ &\quad \tfrac{1}{N}\left(\nabla^2 f_i(\boldsymbol{\theta}^{\tau_{i_k}^{k-1}})\boldsymbol{\theta}^{\tau_{i_k}^{k-1}} - \nabla^2 f_i(\boldsymbol{\theta}^k)\boldsymbol{\theta}^k\right) \\ \boldsymbol{H}^k &:= \boldsymbol{H}^{k-1} + \tfrac{1}{N}\left(\nabla^2 f_{i_k}(\boldsymbol{\theta}^k) - \nabla^2 f_{i_k}(\boldsymbol{\theta}^{\tau_{i_k}^{k-1}})\right) \ .\end{aligned} \tag{8}$$

Leveraging these recursions, we can obtain distributed protocols for implementing SUCAG on G, via transmitting the aggregated (previous) gradients and Hessians. To this end, a key aspect is that the required updates for SUCAG [cf. (3), (7), (8)] are *locally computable* by an activated agent i_k, given the latest estimate $\boldsymbol{\theta}^k$ and aggregate gradient/Hessian information $\boldsymbol{b}^{k-1}, \boldsymbol{H}^{k-1}$.

First, for the special case when G is a *star graph*, a simple protocol can be implemented by using the hub node as a coordinator to store the tuple $\{\boldsymbol{b}^{k-1}, \boldsymbol{H}^{k-1}, \boldsymbol{\theta}^k\}$ and, at each iteration: *(i)* the coordinator activates an agent independently and uniformly at random and passes along this tuple, *(ii)* the agent performs the required updates [cf. (8),(7),(3)] and *(iii)* transmits back the updated values $\{\boldsymbol{b}^k, \boldsymbol{H}^k, \boldsymbol{\theta}^{k+1}\}$. Such protocol is ideal for cloud computing services.

A crucial point of our analysis (cf. Theorem 1) reveals that the requirement of independent and uniform agent activation *can be relaxed*. In fact, it is adopted primarily for demonstration purposes (to ensure unbiasedness, $\mathbb{E}_k[\boldsymbol{g}^k_{\mathsf{SUCAG}}] = \nabla F(\boldsymbol{\theta}^k)$) so as to portray our scheme within the popular stochastic variance reduction framework. This important attribute allows for distributed computations along a random walk on a connected graph G as follows: at iteration k, activated agent i_k performs the update (3) and *(i)* stores $\boldsymbol{\theta}^k$[2], *(ii)* chooses one of its neighbors $i_{k+1} \in \mathcal{N}_{i_k}$ at random as the next active agent, and *(iii)* transmits the updated

[2]In light of storage/computation trade-off, it is also possible for agent i_k to store $\{\boldsymbol{\theta}^{k+1}, \nabla f_{i_k}(\boldsymbol{\theta}^k), \nabla^2 f_{i_k}(\boldsymbol{\theta}^k)\}$ in order to avoid re-evaluation at the next activation instance.

$\{\boldsymbol{b}^k, \boldsymbol{H}^k, \boldsymbol{\theta}^{k+1}\}$ to agent i_{k+1}. This process is illustrated in Fig. 1.

We remark that the above line of reasoning directly applies to *any* stochastic algorithm with the *random selection-aggregate-update* paradigm, e.g., SG, SAG and SAGA; see [9], [10].

III. Convergence Analysis

This section establishes the linear convergence of the SUCAG method with high probability. We begin by stating the required assumptions on the objective function of (1):

A1 *Each component function $f_i(\boldsymbol{\theta})$ has $L_{H,i}$-Lipschitz continuous Hessian. In other words, for any $i \in \{1, ..., N\}$, there exists $L_{H,i} > 0$, such that for all $\boldsymbol{\theta}', \boldsymbol{\theta} \in \mathbb{R}^d$*

$$\|\nabla^2 f_i(\boldsymbol{\theta}) - \nabla^2 f_i(\boldsymbol{\theta}')\| \leq L_{H,i}\|\boldsymbol{\theta} - \boldsymbol{\theta}'\| \; . \tag{9}$$

We define $\bar{L}_H := \max_{i\in\{1,...,N\}} L_{H,i}$.

A2 *The gradient of sum function $F(\boldsymbol{\theta})$ is L-Lipschitz continuous,* i.e., *there exists $L > 0$, such that for all $\boldsymbol{\theta}', \boldsymbol{\theta} \in \mathbb{R}^d$,*

$$\|\nabla F(\boldsymbol{\theta}) - \nabla F(\boldsymbol{\theta}')\| \leq L\|\boldsymbol{\theta} - \boldsymbol{\theta}'\| \; . \tag{10}$$

A3 *The sum function $F(\boldsymbol{\theta})$ is μ-strongly convex, $\mu > 0$,* i.e., *for all $\boldsymbol{\theta}', \boldsymbol{\theta} \in \mathbb{R}^d$,*

$$F(\boldsymbol{\theta}') \geq F(\boldsymbol{\theta}) + \langle \nabla F(\boldsymbol{\theta}), \boldsymbol{\theta}' - \boldsymbol{\theta}\rangle + \frac{\mu}{2}\|\boldsymbol{\theta}' - \boldsymbol{\theta}\|^2 \; . \tag{11}$$

Under A3, a unique optimal solution to problem (1) exists and it is denoted by $\boldsymbol{\theta}^\star$. We denote the condition number for $F(\boldsymbol{\theta})$ by $\kappa(F) := L/\mu$. Note that the above are standard assumptions that can be satisfied in many applications; for example, ℓ_2−regularized logistic regression [5], and quadratic programming such as (overdetermined) least-squares and Model Predictive Control (MPC) [26] applications.

We also state the following assumption on the delays in the SUCAG method:

A4 *For all $k \geq 1$, it holds that $|k - \tau_i^{k-1}| \leq m_k$ for all i, where $\{m_k\}_{k\geq 1}$ is non-decreasing with $m_1 \geq 1$. Moreover, it holds that*

$$4 \log m_k \leq c_0 + \log k \; , \tag{12}$$

for all $k \geq 1$ and some $c_0 \geq 0$.

Note that it follows that A4 implies that $\{m_k\}_{k\geq 1}$ satisfies

$$2m_k \leq m_0 + \big((1/3) - \beta\big)k, \tag{13}$$

for proper selection of $m_0 > 0$ and $\beta \in (0, \frac{1}{3})$. In the interest of space, the detailed proofs for the below analysis can be found in our online appendix (`https://arxiv.org/abs/1803.08198`).

Later, we will show that the distributed message passing protocols in Section II-A satisfies A4 with high probability. The following establishes the linear convergence of SUCAG:

Theorem 1 *Assume that A1, A2, A3, A4 hold, and let*[3] *$\mu \geq 1$. Fix $\delta = 1 - \Delta\gamma\frac{2\mu L}{\mu+L}$ for some sufficiently small $\Delta \in (0, 1)$, and step size γ that satisfies:*

$$\begin{aligned} &\gamma \leq \frac{2}{\mu + L} \\ &\gamma \leq \min_{j=1,...,4} \Big(\frac{3m_0}{2}\frac{\mu L}{\mu + L}\Delta + \frac{1}{C'_j}\Big)^{-1} \; , \end{aligned} \tag{14}$$

where the constants are given by:

$$C'_j := \frac{e^{1-c_0}\beta\Delta(1-\Delta)}{4\tilde{q}_j\|\boldsymbol{\theta}^0 - \boldsymbol{\theta}^\star\|^{2\tilde{\eta}_j - 2}}\Big(\frac{2\mu L}{\mu + L}\Big)^2, \; j = 1, ..., 4 \; , \tag{15}$$

with

$\tilde{q}_1 := 2\bar{L}_H L^2, \tilde{q}_2 := 32\bar{L}_H^3, \tilde{q}_3 := 8\bar{L}_H^2 L^4, \tilde{q}_4 := 2048\bar{L}_H^6,$
$\tilde{\eta}_1 := 3/2, \; \tilde{\eta}_2 := 5/2, \; \tilde{\eta}_3 := 2, \; \tilde{\eta}_4 := 4 \; .$

Then, the following properties hold:

1) *The SUCAG method converges linearly as $\|\boldsymbol{\theta}^k - \boldsymbol{\theta}^\star\|^2 \leq \delta^k\|\boldsymbol{\theta}^0 - \boldsymbol{\theta}^\star\|^2$ for all $k \geq 0$.*
2) *There exists an upper bound sequence $\{\bar{V}(k)\}_{k\geq 0}$ such that $\|\boldsymbol{\theta}^k - \boldsymbol{\theta}^\star\|^2 \leq \bar{V}(k)$ for all $k \geq 0$ with the rate:*

$$\lim_{k\to\infty} \frac{\bar{V}(k+1)}{\bar{V}(k)} = 1 - 2\gamma\frac{\mu L}{\mu + L} \; . \tag{16}$$

First, from Theorem 1, we consider the case when $\|\boldsymbol{\theta}^0 - \boldsymbol{\theta}^\star\| \approx 0$. In particular, taking $\Delta = \|\boldsymbol{\theta}^0 - \boldsymbol{\theta}^\star\|^{\frac{1}{2}}$ shows that the right hand side of the second inequality in (14) is at the order of $\Omega(\|\boldsymbol{\theta}^0 - \boldsymbol{\theta}^\star\|^{-\frac{1}{2}})$. When $\|\boldsymbol{\theta}^0 - \boldsymbol{\theta}^\star\| \approx 0$, the SUCAG method is allowed to take the step size at $\gamma = 2/(\mu + L)$ in (14). Substituting this into (16) shows that the asymptotic linear convergence rate of the SUCAG method is $1 - 4\mu L/(\mu+L)^2 = 1 - \Omega(1/\kappa(F))$. Consequently, the method finds an ϵ-optimal solution to (1) using just $\mathcal{O}(\kappa(F)\log(1/\epsilon))$ steps.

Second, assumption A4 on the delays in the SUCAG method allows for a sub-linear delay which grows *unboundedly* as fast as $m_k = \mathcal{O}(k^{1/4})$. This is of relevance to the two distributed implementations discussed in Section II-A. In particular, in both cases the agents' activations can be captured by a finite state Markov chain, and the time delays can be analyzed via the *hitting time* of the chain:

Lemma 1 *Consider the sequence $(i_k)_{k\geq 0}$ that is governed by a finite, irreducible and aperiodic Markov chain, and suppose that the chain is stationary*[4]. *It holds that*

$$P(|k - \tau_i^{k-1}| > x) \leq \exp\Big(-\frac{x}{1 + e\bar{\tau}_i} + 1\Big), \; \forall \; x \geq 0, \tag{17}$$

for all $i = 1, ..., N$, where $\bar{\tau}_i$ is the expected first visit time. In particular, the delay $|k - \tau_i^{k-1}|$ can be bounded by $\mathcal{O}(\bar{\tau}_i + \bar{\tau}_i \log(Nk^2/\epsilon)$ with probability at least $1 - \epsilon$.

In specific, for any connected undirected graph, the Markov chain is stationary and $\bar{\tau}_i = N$ provided that the first activated agent is selected uniformly at random. Consequently,

[3]This is with absolutely no loss in generality, and is solely used in the analysis so as to simply notation in the derived stepsize selection rule.

[4]We note that the analysis applies even to non stationary Markov chains; we skip the details for length considerations.

even though $m_k \to \infty$ as $k \to \infty$, the time varying delay m_k satisfies assumption A4 w.h.p.

Importantly, combining Theorem 1 and Lemma 1 shows that when the initialization is sufficiently close to the optimal solution, w.h.p. the distributed implementation of SUCAG converges linearly at an asymptotic rate that is *independent from the communication graph structure*.

A. Proof of Theorem 1

The proof consists of two parts — first we prove a descent lemma for the SUCAG method to characterize its per-iteration progress; then we show that the resulting nonlinear process converges linearly, with asymptotic rate as in (16). Let us state the first result as follows:

Lemma 2 *Assume that A1, A2, A3, A4 hold and $\gamma \leq 2/(\mu+L)$. The following holds for all $k \geq 0$:*

$$\begin{aligned}\|\boldsymbol{\theta}^{k+1} - \boldsymbol{\theta}^\star\|^2 &\leq \Big(1 - 2\gamma\frac{\mu L}{\mu+L}\Big)\|\boldsymbol{\theta}^k - \boldsymbol{\theta}^\star\|^2 \\ &+ \gamma^3(m_k)^2 \max_{(k-2m_k)\leq \ell \leq k}\Big(\tilde{q}_1\|\boldsymbol{\theta}^\ell - \boldsymbol{\theta}^\star\|^3 + \tilde{q}_2\|\boldsymbol{\theta}^\ell - \boldsymbol{\theta}^\star\|^5\Big) \\ &+ \gamma^6(m_k)^4 \max_{(k-2m_k)\leq \ell \leq k}\Big(\tilde{q}_3\|\boldsymbol{\theta}^\ell - \boldsymbol{\theta}^\star\|^4 + \tilde{q}_4\|\boldsymbol{\theta}^\ell - \boldsymbol{\theta}^\star\|^8\Big)\end{aligned} \tag{18}$$

An important fact observed is that the distance to optimal solution at the $(k+1)-$th iteration follows an inequality system with high order terms of the distance to optimal solution evaluated at *delayed* time instances.

Let $R(k) := \|\boldsymbol{\theta}^k - \boldsymbol{\theta}^\star\|^2$, the above motivates us to study the system:

$$R(k+1) \leq pR(k) + \sum_{j=1}^{J} q_j m_{1,j}^{(k)} \max_{(k-m_{2,j}^{(k)})_+ \leq \ell \leq k} R^{\eta_j}(\ell), \tag{19}$$

where $p \in (0,1)$, $\eta_j > 1$ for all j and $R(k)$ is non-negative. Moreover, we have $m_{1,j}^{(k)}, m_{2,j}^{(k)} \geq 0$. The following lemma establishes the linear convergence for $R(k)$ under a sufficient condition:

Lemma 3 *Fix $\delta \in (0,1)$ such that $\delta > p$. Assume that*

$$\log(m_{1,j}^{(k)}) \leq c_0 + c_1 \log k,\ m_{2,j}^{(k)} \leq m_0 + \Big(\frac{\eta-1}{\eta} - \beta\Big)k\ , \tag{20}$$

for some $m_0, c_0, c_1 \geq 1$, $0 < \beta < (\eta-1)/\eta$, and $\eta := \min_{j=1,\ldots,J}\eta_j > 1$. Moreover, assume that

$$q_j \leq \frac{1}{R^{\eta_j-1}(0)}\frac{\delta^{-\xi_j^\star(\delta)}(\delta-p)}{J}\ ,\ j = 1,\ldots,J\ , \tag{21}$$

with the non-positive number $\xi_j^\star(\delta)$ defined as:

$$\xi_j^\star(\delta) := \min_{k\geq 0}\Big(\frac{\log m_{1,j}^{(k)}}{\log \delta} + \eta_j\big(k - m_{2,j}^{(k)}\big)_+ - k\Big)\ . \tag{22}$$

It holds for (19) *that* (i) *$R(k) \leq \delta^k R(0)$ for all $k \geq 0$, and* (ii) *there exists an upper bound sequence $\{\bar{R}(k)\}_{k\geq 0}$ such that $\bar{R}(k) \geq R(k)$ for all $k \geq 0$ and its convergence rate is:*

$$\lim_{k\to\infty} \bar{R}(k+1)/\bar{R}(k) = p\ . \tag{23}$$

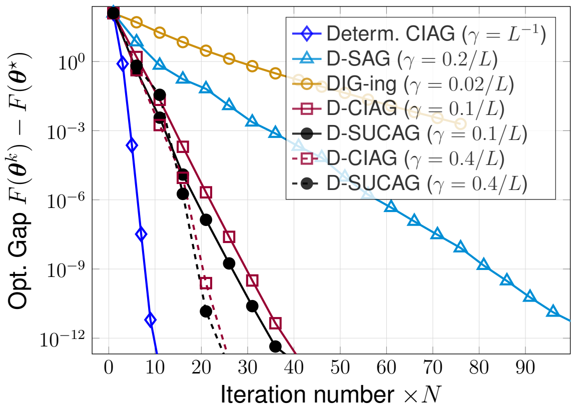

Fig. 2: Comparison of distributed optimization methods for solving the logistic regression problem (24) over a time varying graph with $N = 250$ agents, batch size $B = 1$, and $d = 51$ parameters. The optimality gap against iteration number is obtained through averaging over 10 trials (except for the deterministic CIAG method).

We remark that a crucial point to guaranteeing the linear convergence rate is on the high order terms with power $\eta_j > 1$. In fact, for $\eta_j = 1$ with some j, it can be shown that $R(k)$ may diverge as $m_{1,j}^{(k)}, m_{2,j}^{(k)}$ is growing sublinearly in (20).

Finally, we observe that (18) is a special case of (19) by substituting $R(k) = \|\boldsymbol{\theta}^k - \boldsymbol{\theta}^\star\|^2$ and the parameters:

$$\begin{aligned}&p = 1 - \frac{2\gamma\mu L}{\mu+L},\ \eta_1 = \frac{3}{2},\ \eta_2 = \frac{5}{2},\ \eta_3 = 2,\ \eta_4 = 4\ ,\\ &q_1 = \gamma^3\tilde{q}_1,\ q_2 = \gamma^3\tilde{q}_2,\ q_3 = \gamma^6\tilde{q}_3,\ q_4 = \gamma^6\tilde{q}_4\ ,\\ &m_{1,j}^{(k)} = (m_k)^{2\lceil \frac{j}{2}\rceil},\ m_{2,j}^{(k)} = 2m_k,\ j = 1,\ldots,4\ ,\end{aligned}$$

Under A4, we observe that the above $m_{1,j}^{(k)}, m_{2,j}^{(k)}$ satisfy (20) in Lemma 3 with the same c_0 . Moreover, in the online appendix we show that the step size γ specified in (14) gives the set of q_j that satisfy (21). This concludes the proof.

IV. Numerical Experiments

We evaluate the performance SUCAG on the learning task of training a linear classifier via logistic regression. In this problem, the $i-$th component function is defined as:

$$f_i(\boldsymbol{\theta}) := \sum_{b=1}^{B}\log(1+\exp(-y_{i,b}\langle\boldsymbol{\theta}, \boldsymbol{x}_{i,b}\rangle)) + \frac{B}{2N}\|\boldsymbol{\theta}\|^2\ , \tag{24}$$

where $(\boldsymbol{x}_{i,b}, \boldsymbol{y}_{i,b})_{b=1}^B$ represents data held by agent i, and B is the size of the corresponding data subset. It can be verified that (24) and its corresponding sum function $F(\boldsymbol{\theta})$ satisfy A1–A3. We generate a synthetic dataset in the experiment where the ground truth classifier $\boldsymbol{\theta}_0$ is selected as $\boldsymbol{\theta}_0 \sim \mathcal{U}[-1,1]^d$, then each feature vector $\boldsymbol{x}_i$ is generated independently following $\boldsymbol{x}_i \sim \mathcal{U}[-1,1]^d$ with the label y_i computed as $y_i = \text{sign}(\langle\boldsymbol{x}_i, \boldsymbol{\theta}_0\rangle)$. To simulate the distributed optimization methods, we generate G as an Erdos-Renyi graph with N agents and a connection probability of $2\frac{\log N}{N}$ (where, if needed, we repeat the procedure until a connected graph is obtained). The SUCAG method operates on the aforementioned graph using the protocol in Section II-A (see also Fig.1), where the probability for selecting the next agent is uniform over all neighbors. For benchmark purposes, we compare the SUCAG method to the similarly modified version of distributed CIAG [17] and SAG [13]. Moreover, we compare the DIG-ing method [27], an average consensus-based distributed optimization method that accounts for time varying communication topologies. For this method, the mixing matrix is designed in a similar fashion as the pairwise gossip protocol [28], while the agents' activation sequence follows the same model as in our proposed scheme. The deterministic CIAG method [17] implemented on a fixed ring graph is also compared for benchmarking purpose.

Fig. 2 shows the numerical findings (averaged over 10 generated random walks) for a synthetic dataset example with $d = 51$ parameters, $N = 250$ agents, where each agent has a batch of $B = 1$ data tuple, *i.e.,* there are $NB = 250$ data tuples in total. For both SAG and DIG-ing methods, we have optimized their step sizes (as $\gamma = 0.2/L$ and $\gamma = 0.02/L$, respectively) used so that the methods converge in the trials performed. For SUCAG and distributed CIAG methods, two step size configurations are compared. The numerical results show that the SUCAG method (as well as the distributed CIAG method) outperform all other methods, and thus clearly demonstrating the benefits of curvature-aided gradient tracking in accelerating distributed optimization.

V. Conclusions

We have proposed an unbiased curvature-aided stochastic gradient method for large-scale optimization. Our method is implementable in a distributed setting, both via distributed computations performed from agents activated by a single coordinator, as well as via a fully distributed random walk on the communication graph. Under usual assumptions, our convergence analysis establishes linear convergence at a rate that solely depends on the condition number, but is *independent from the network size* provided that the initialization is sufficiently close to optimality. Besides, we have established more general results under rather weak requirements, *i.e.,* infinitely often activation, even when activation delays grow unbounded: our line of reasoning may be applied to a wide class of stochastic optimization primitives. Our numerical experiments verified faster convergence compared to various methods for distributed and stochastic optimization.

Acknowledgements. This work was supported by NSF under grants NSF CCF-BSF 1714672, CCF-1717207 and CCF-1717391. The authors would like to thank Prof. Maxim Raginsky for pointing out the reference [29].

References

[1] A. Nedić and A. Ozdaglar, "Distributed subgradient methods for multi-agent optimization," *IEEE Transactions on Automatic Control*, vol. 54, no. 1, pp. 48–61, 2009.

[2] X. Lin, N. B. Shroff, and R. Srikant, "A tutorial on cross-layer optimization in wireless networks," *IEEE Journal on Selected areas in Communications*, vol. 24, no. 8, pp. 1452–1463, 2006.

[3] N. Freris, H. Kowshik, and P. R. Kumar, "Fundamentals of large sensor networks: Connectivity, capacity, clocks, and computation," *Proceedings of the IEEE*, vol. 98, no. 11, pp. 1828–1846, 2010.

[4] S. Boyd, N. Parikh, E. Chu, B. Peleato, J. Eckstein *et al.*, "Distributed optimization and statistical learning via the alternating direction method of multipliers," *Foundations and Trends® in Machine learning*, vol. 3, no. 1, pp. 1–122, 2011.

[5] V. Vapnik, "An overview of statistical learning theory," *IEEE Transactions on Neural Networks*, vol. 10, no. 5, pp. 988–999, 1999.

[6] J. Tsitsiklis, "Problems in decentralized decision making and computation," Ph.D. dissertation, Dept. of Electrical Engineering and Computer Science, M.I.T., Boston, MA, 1984.

[7] A. Nedić, D. P. Bertsekas, and V. S. Borkar, "Distributed asynchronous incremental subgradient methods," *Studies in Computational Mathematics*, vol. 8, no. C, pp. 381–407, 2001.

[8] K. Scaman, F. Bach, S. Bubeck, Y. T. Lee, and L. Massoulié, "Optimal algorithms for smooth and strongly convex distributed optimization in networks," *arXiv preprint arXiv:1702.08704*, 2017.

[9] B. Johansson, M. Rabi, and M. Johansson, "A randomized incremental subgradient method for distributed optimization in networked systems," *SIAM Journal on Optimization*, vol. 20, no. 3, pp. 1157–1170, 2009.

[10] S. S. Ram, A. Nedić, and V. V. Veeravalli, "Incremental stochastic subgradient algorithms for convex optimization," *SIAM Journal on Optimization*, vol. 20, no. 2, pp. 691–717, 2009.

[11] N. Freris and A. Zouzias, "Fast distributed smoothing of relative measurements," in *Proc. CDC*, 2012, pp. 1411–1416.

[12] H. Robbins and S. Monro, "A stochastic approximation method," *The annals of mathematical statistics*, pp. 400–407, 1951.

[13] M. Schmidt, N. Le Roux, and F. Bach, "Minimizing finite sums with the stochastic average gradient," *Mathematical Programming*, vol. 162, no. 1-2, pp. 83–112, 2017.

[14] A. Defazio, F. Bach, and S. Lacoste-Julien, "SAGA: A fast incremental gradient method with support for non-strongly convex composite objectives," in *NIPS*, 2014, pp. 1646–1654.

[15] R. Johnson and T. Zhang, "Accelerating stochastic gradient descent using predictive variance reduction," in *NIPS*, 2013, pp. 315–323.

[16] A. Nedić and D. P. Bertsekas, "Incremental subgradient methods for nondifferentiable optimization," *SIAM Journal on Optimization*, vol. 12, no. 1, pp. 109–138, 2001.

[17] H.-T. Wai, W. Shi, A. Nedić, and A. Scaglione, "Curvature-aided incremental aggregated gradient method," in *Proc. Allerton*, 2017.

[18] D. P. Bertsekas and J. Tsitsiklis, *Parallel and distributed computation: numerical methods*. Prentice hall Englewood Cliffs, NJ, 1989, vol. 23.

[19] R. M. Gower, N. L. Roux, and F. Bach, "Tracking the gradients using the Hessian: A new look at variance reducing stochastic methods," *arXiv preprint arXiv:1710.07462*, 2017.

[20] N. Freris, S. Graham, and P. Kumar, "Fundamental limits on synchronizing clocks over networks," *IEEE Transactions on Automatic Control*, vol. 56, no. 6, pp. 1352–1364, 2011.

[21] K.-D. Kim and P. R. Kumar, "Cyber–physical systems: A perspective at the centennial," *Proceedings of the IEEE*, vol. 100, no. Special Centennial Issue, pp. 1287–1308, 2012.

[22] N. Freris, O. Oçal, and M. Vetterli, "Compressed sensing of streaming data," in *Proc. Allerton*, 2013, pp. 1242–1249.

[23] P. Sopasakis, N. Freris, and P. Patrinos, "Accelerated reconstruction of a compressively sampled data stream," in *Proc. EUSIPCO*, 2016, pp. 1078–1082.

[24] E. Moulines and F. R. Bach, "Non-asymptotic analysis of stochastic approximation algorithms for machine learning," in *NIPS*, 2011, pp. 451–459.

[25] Y. Nesterov, "Introductory lectures on convex programming volume I: Basic course," *Lecture notes*, 1998.

[26] M. Morari and J. H. Lee, "Model predictive control: past, present and future," *Computers & Chemical Engineering*, vol. 23, no. 4-5, pp. 667–682, 1999.

[27] A. Nedić, A. Olshevsky, and W. Shi, "Achieving geometric convergence for distributed optimization over time-varying graphs," *SIAM Journal on Optimization*, vol. 27, no. 4, pp. 2597–2633, 2017.

[28] S. Boyd, A. Ghosh, B. Prabhakar, and D. Shah, "Randomized gossip algorithms," *IEEE transactions on information theory*, vol. 52, no. 6, pp. 2508–2530, 2006.

[29] S. Roch, "Modern discrete probability: An essential toolkit," *Lecture notes*, 2015.