Limited circulation. For review only
IEEE-TAC Submission no.: 23-1500.1

Set-wise Coordinate Descent for Dual Asynchronous Decentralized Optimization

Marina Costantini, Nikolaos Liakopoulos, Panayotis Mertikopoulos, and Thrasyvoulos Spyropoulos

***Abstract*— In decentralized optimization over networks, synchronizing the updates of all nodes incurs significant communication overhead. For this reason, much of the recent literature has focused on the analysis and design of asynchronous optimization algorithms where nodes can activate anytime and contact a single neighbor to complete an iteration together. However, most works assume that the neighbor selection is done at random based on a fixed probability distribution (e.g., uniform), a choice that ignores the optimization landscape at the moment of activation. Instead, in this work we introduce an optimization-aware selection rule that chooses the neighbor providing the highest dual cost improvement (a quantity related to a dualization of the problem based on consensus). This scheme is related to the coordinate descent (CD) method with the Gauss-Southwell (GS) rule for coordinate updates; in our setting however, only a subset of coordinates is accessible at each iteration (because each node can communicate only with its neighbors), so the existing literature on GS methods does not apply. To overcome this difficulty, we develop a new analytical framework for smooth and strongly convex functions that covers our new class of *set-wise CD algorithms* –a class that applies to both decentralized and parallel distributed computing scenarios– and we show that the proposed set-wise GS rule can speed up the convergence in terms of iterations by a factor equal to the size of the largest coordinate set. We analyze extensions of these algorithms that exploit the knowledge of the smoothness constants when available and otherwise propose an algorithm to estimate these constants. Finally, we validate our theoretical results through extensive simulations.**

***Index Terms*— convex optimization, coordinate descent, decentralized optimization, distributed machine learning, distributed optimization, multi-agent optimization, optimization over networks.**

I. Introduction

Many timely applications require solving optimization problems over a network where nodes can only communicate with their direct neighbors. This may be due to the need of distributing storage and computation loads (e.g. training large machine learning models [1]), or to avoid transferring data that is naturally collected in a decentralized manner, either due to the communication costs or to privacy reasons (e.g. sensor networks [2], edge computing [3], and precision medicine [4]).

Specifically, we consider a setting where the nodes want to solve the decentralized optimization problem

$$\underset{\theta\in\mathbb{R}^d}{\text{minimize}} \quad \sum_{i=1}^{n} f_i(\theta), \tag{1}$$

where each local function f_i is known only by node i and nodes can exchange optimization values (parameters, gradients) but *not* the local functions themselves. We represent the communication network as a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with $n = |\mathcal{V}|$ nodes (agents) and $E = |\mathcal{E}|$ edges, which are the links used by the nodes to communicate with their neighbors.

Existing methods to solve this problem usually assign to each node a local variable θ_i and execute updates that interleave a local gradient step at the nodes followed by an averaging step (usually carried out by a doubly-stochastic matrix) that aggregates the update of the node with those of its neighbors [5]–[10].

Alternatively, the consensus constraint can be stated explicitly between node pairs connected by an edge:

$$\underset{\theta_1,\dots,\theta_n\in R^d}{\text{minimize}} \quad \sum_{i=1}^{n} f_i(\theta_i) \tag{2a}$$

$$\text{subject to} \quad \theta_i = \theta_j \quad \forall\ (i,j) \equiv \ell \in \mathcal{E}, \tag{2b}$$

where $\ell \equiv (i,j)$ indicates that edge ℓ links nodes i and j. This reformulation allows solving the decentralized optimization problem through its dual problem. Interestingly, dual decentralized algorithms have been the only ones shown to achieve optimal convergence rates when using acceleration [11]–[13]. However, these algorithms are *synchronous*, in the sense that they require coordinating the updates of all nodes at each iteration. In contrast, by exploiting connections with coordinate descent theory naturally derived from the dual formulation, here we propose an alternative way to speed up convergence in dual decentralized optimization that does *not* require network-wide synchronicity.

In our setup, nodes activate anytime at random and select one of their neighbors to make an update together. Methods with such minimal coordination requirements avoid incurring extra costs of synchronization that may also slow down convergence [14]–[18]. However, most of these works assume that when a node activates, it simply selects the neighbor to contact randomly, based on a predefined probability distribution. This approach overlooks the possibility of letting nodes *choose* the neighbor to contact taking into account the optimization

Paper submitted for review on August 17th, 2023.
M. Costantini is with EURECOM, Sophia Antipolis, France (costanti@eurecom.fr).
N. Liakopoulos is with Amazon, Luxembourg City, Luxembourg (nliako@amazon.lu).
P. Mertikopoulos is with Univ. Grenoble Alpes, CNRS, Inria, Grenoble INP, & LIG, France (panayotis.mertikopoulos@imag.fr).
T. Spyropoulos is with EURECOM & the Technical University of Crete, Chania, Greece (spyropou@eurecom.fr).

landscape at the time of activation. Therefore, here we depart from the probabilistic choice and ask: *can nodes pick the neighbor smartly to make the optimization converge faster?*

In this paper, we give an affirmative answer and propose algorithms that achieve this by solving the dual problem of (2). In the dual formulation, there is one dual variable $\lambda_\ell \in \mathbb{R}^d$ per constraint $\theta_i = \theta_j$, hence each dual variable can be associated with an edge ℓ in the graph. Our algorithms let an activated node i contact a neighbor j so that together they update their shared variable λ_ℓ, $\ell \equiv (i,j)$ with a gradient step. In particular, we propose to select the neighbor j such that the updated λ_ℓ is the one *whose directional gradient for the dual function is the largest*, and thus the one that provides the greatest cost improvement at that iteration. Beyond the preliminary version of this work that appeared in [19], such optimal choice has not yet been considered in the literature.

Interestingly, the above protocol where a node activates and selects a λ_ℓ to update can be seen as applying the coordinate descent (CD) method [20] to solve the dual problem of (2), with the following key difference: unlike standard CD methods, where *any* of the coordinates may be updated, now *only a small subset of coordinates are accessible at each step*, which are the coordinates associated with the edges connected to the node activated. Moreover, our proposal of updating the λ_ℓ with the largest gradient is similar to the Gauss-Southwell (GS) rule [21], but applied *only* to the parameters accessible by the activated node.

We name such protocols *set-wise CD* algorithms, and we analyze a number of possibilities for the coordinate (or equivalently, neighbor) choice: random sampling, local GS, and two extensions that take into account the smoothness constants of the dual function. Compared to standard CD literature, three difficulties complicate the analysis and constitute the base of our contributions: (i) for arbitrary graphs, the dual problem of (2) has an objective function that is *not* strongly convex, even if the primal functions f_i *are* strongly convex, (ii) the fact that the GS rule is applied to a few coordinates prevents the use of standard norms to obtain the linear rate, as commonly done for CD methods [20]–[22], and (iii) the coordinate sets are overlapping (i.e. non-disjoint), which makes the problem even harder.

Our results also apply to the (primal) parallel distributed setting where multiple workers modify different sets of coordinates of the parameter vector, which is stored in a server accessible by all workers [23]–[25]. In particular, we show that for this setting the GS selection attains the maximum speedup promised by the theory (see Theorem 2).

Our contributions can be summarized as follows:

- We introduce the class of *set-wise CD* algorithms for asynchronous optimization applicable to both the decentralized and the parallel distributed settings.
- We prove linear convergence rates for smooth and strongly convex f_i and four coordinate (equivalently, neighbor) choices: random uniform, GS, and their variants when the coordinate smoothness constants are known.
- For the cases when these constants are not known, we propose an algorithm based on backtracking [20] for estimating them online, and show that for certain problems this method achieves faster per-iteration convergence than the exact knowledge of these constants.
- To obtain the rates of all considered algorithms, we prove strong convexity in uniquely-defined norms that (i) take into account the graph structure to show strong convexity in the linear subspace where the coordinate updates are applied, and (ii) account for both the random uniform node activation and the application of the GS rule to just a subset of the coordinates.
- We show that the speedup in terms of number of iterations of GS selection with respect to random uniform can be up to $N_{\max}$ (the size of the largest coordinate set).
- We prove that the versions accounting for coordinate smoothness are provably faster than those that do not account for these constants. In particular, we show that the algorithm exploiting both the GS rule *and* the smoothness knowledge is provably faster than all others.
- We support all our results with thorough simulations[1].

II. Related work

A number of algorithms have been proposed to solve (1) asynchronously. In [18], the activated node chooses a neighbor uniformly at random and both nodes average their primal local values. In [14] the authors adapted the ADMM algorithm to the decentralized setting, but it was the ADMM of [15] the first one shown to converge at the same rate as the centralized ADMM. The algorithm of [16] tracks the average gradients to converge to the exact optimum instead of just a neighborhood around it, as many algorithms back then. The algorithm of [26] can be used on top of directed graphs, which impose additional challenges. A key novelty of our scheme, compared to this line of work, is that we consider the possibility of letting the nodes *choose smartly* the neighbor to contact in order to make convergence faster.

Work [27] is, to the best of our knowledge, the only work similarly considering smart neighbor selection. The authors propose Max-gossip, a version of the (primal) algorithm in [5] where the activated node averages its local parameter with that of the neighbor with whom the parameter difference is the largest. They show that the algorithm converges sublineraly to the optimum (for convex functions), and show in numerical simulations that it outperforms random neighbor selection. In contrast, here we propose dual algorithms for which we show linear convergence rates (for smooth and strongly convex functions), and most importantly, (i) we *prove analytically* that either applying the GS rule and/or using the Lipschitz information achieves faster convergence than random neighbor sampling, and (ii) we *quantify* the magnitude of the gains.

Finally, as mentioned earlier, our work relates to standard CD literature. In particular, our theorems extend the results in [21], where the GS rule was shown to be up to d times faster than uniform sampling for $f\colon \mathbb{R}^d \to \mathbb{R}$, to the case where this choice is constrained to a subset of the coordinates only, sets have different sizes, each coordinate belongs to

[1]The code to reproduce the results is available at `https://github.com/m-costantini/Set-wise_Coordinate_Descent/`.

exactly two sets, and sets activate uniformly at random. As we explain in Sections IV and V, these considerations bring important new challenges with respect to the standard single-machine CD algorithms. Furthermore, our algorithms are not only applicable to the decentralized case but also to parallel distributed settings such as [23]–[25]. For the latter, [28] also analyzed the GS applied to coordinate subsets, but their sets are disjoint, accessible by any worker, and they do not quantify the speedup of the method with respect to random uniform sampling.

III. Dual formulation

In this section, we define the notation, obtain the dual problem of (2), and analyze the properties of the dual objective function. We will assume throughout that the functions f_i are M_i-smooth and μ_i-strongly convex, i.e. there exist finite constants $M_i \geq \mu_i > 0, i \in [n]$ such that:

$$f_i(y) \leq f_i(x) + \langle \nabla f(x), y - x \rangle + (M_i/2) \|y - x\|_2^2$$
$$f_i(y) \geq f_i(x) + \langle \nabla f(x), y - x \rangle + (\mu_i/2) \|y - x\|_2^2 .$$

We define the concatenated primal and dual variables $\theta = [\theta_1^T, \ldots, \theta_n^T]^T \in \mathbb{R}^{nd}$ and $\lambda = [\lambda_1^T, \ldots, \lambda_E^T]^T \in \mathbb{R}^{Ed}$, respectively. The graph's incidence matrix $A \in \mathbb{R}^{n \times E}$ has exactly one 1 and one -1 per column ℓ, in the rows corresponding to nodes $i, j : \ell \equiv (i, j)$, and zeros elsewhere (the choice of sign for each node is irrelevant). We call $u_i \in \mathbb{R}^n$ the vector that has 1 in entry i and 0 elsewhere; we define $e_\ell \in \mathbb{R}^E$ analogously. We use $k \in [K]$ to indicate $k = 1, \ldots, K$. Vectors $\mathbf{1}$ and $\mathbf{0}$ are respectively the all-one and all-zero vectors, and I_d is the $d \times d$ identity matrix. Finally, in order to use matrix operations in the equations below for some operations, we define the block arrays $\Lambda = A \otimes I_d \in \mathbb{R}^{nd \times Ed}$ and $U_i = u_i \otimes I_d \in \mathbb{R}^{nd \times d}$, where $\otimes$ is the Kronecker product. This operation generates arrays analogous to A and u_i where the original entries 1, -1, and 0 have been replaced by I_d, $-I_d$, and the all-zero $d \times d$ matrix, respectively.

We can rewrite now (2b) as $\Lambda^T \theta = \mathbf{0}$, and the node variables as $\theta_i = U_i^T \theta$. The minimum value of (2) satisfies:

$$\begin{aligned}
\inf_{\theta : \Lambda^T \theta = \mathbf{0}} \sum_{i=1}^n f_i(U_i^T \theta) &\overset{(a)}{=} \inf_\theta \sup_\lambda \left[\sum_{i=1}^n f_i(U_i^T \theta) - \lambda^T \Lambda^T \theta \right] \\
&\overset{(b)}{=} \sup_\lambda \inf_\theta \left[\sum_{i=1}^n f_i(U_i^T \theta) - \lambda^T \Lambda^T \theta \right] \\
&= - \inf_\lambda \sup_\theta \sum_{i=1}^n \left[(U_i^T \Lambda \lambda)^T U_i^T \theta - f_i(U_i^T \theta) \right] \\
&= - \inf_\lambda \sum_{i=1}^n f_i^*(U_i^T \Lambda \lambda) \triangleq - \inf_\lambda F(\lambda), \qquad (3)
\end{aligned}$$

where (a) holds due to Lagrange duality and (b) holds by strong duality (see e.g. Sec. 5.4 in [29]). Functions f_i^* are the Fenchel conjugates of the f_i, and are defined as

$$f_i^*(y) = \sup_{x \in \mathbb{R}^d} \left(y^T x - f_i(x) \right).$$

Our set-wise CD algorithms converge to the optimal solution of (2) by solving (3). In particular, they update a single dual variable $\lambda_\ell, \ell \in [E]$ at each iteration and converge to some minimum value λ^* of $F(\lambda)$.

Since $\sum_{i=1}^n f_i(U_i^T \theta)$ in (2a) is $M_{\max}$-smooth and $\mu_{\min}$-strongly convex in θ, with $M_{\max} = \max_i M_i$ and $\mu_{\min} = \min_i \mu_i$, function F is L-smooth with $L = \frac{\gamma_{\max}}{\mu_{\min}}$, where $\gamma_{\max}$ is the largest eigenvalue of $\Lambda^+ \Lambda$ (Sec. 4 in [13]). We call $\gamma_{\min}^+$ the smallest non-zero eigenvalue[2] of $\Lambda^+ \Lambda$.

However, as shown next, function F is *not* strongly convex in the standard L2 norm, which is the property that usually facilitates obtaining linear rates in optimization literature.

Lemma 1. F is not strongly convex in $\|\cdot\|_2$.

Proof. Since Λ does not have full column rank in the general case (i.e., unless the graph is a tree), there exist $w \in \mathbb{R}^{Ed}$ such that $w \neq \mathbf{0}$ and $F(\lambda) = F(\lambda + tw)\ \forall t \in \mathbb{R}$. □

Nevertheless, we can still show linear rates for the set-wise CD algorithms using the following result.

Lemma 2 (*Appendix C of* [12]). F is σ_A-strongly convex in the semi-norm $\|x\|_A \triangleq (x^T \Lambda^+ \Lambda x)^{\frac{1}{2}}$, with $\sigma_A = \frac{\gamma_{\min}^+}{M_{\max}}$.

Above, Λ^+ denotes the pseudo-inverse of Λ. A key fact for the proofs in the next section is that matrix $\Lambda^+ \Lambda$ is a projector onto range(Λ^T), the column space of Λ^T.

To simplify the notation, in what follows we assume that $d = 1$, so that $\Lambda = A$, $U_i = u_i$, and the gradient $\nabla_\ell F(\lambda) = \frac{\partial F(\lambda)}{\partial \lambda_\ell}$ of $F(\lambda)$ in the direction of λ_ℓ is a scalar. In Sec. VI-C we discuss how to adapt our proofs to the case $d > 1$.

IV. Set-wise Coordinate Descent Algorithms

In this section, we present the *set-wise CD* algorithms, which can solve generic convex problems (and (3) in particular) optimally and asynchronously. Here we analyze two possibilities for the coordinate choice within the accessible coordinate subset: (i) sampling uniformly at random (SU-CD), and (ii) applying the GS rule (SGS-CD). Their extensions when the coordinate-wise Lipschitz constants are known (or can be estimated) are analyzed in Section V.

If coordinate ℓ is updated at iteration k and assuming $d = 1$, the standard CD update applied to $F(\lambda)$ is [20]:

$$\lambda^{k+1} = \lambda^k - \eta^k \nabla_\ell F(\lambda^k) e_\ell, \qquad (4)$$

where η^k is the stepsize. Since F is L-smooth, choosing $\eta^k = 1/L\ \forall k$ guarantees descent at each iteration [21]:

$$F(\lambda^{k+1}) \leq F(\lambda^k) - \frac{1}{2L} \left(\nabla_\ell F(\lambda^k) \right)^2 . \qquad (5)$$

Eq. (5) will be the departure point to prove the linear convergence rates of SU-CD and SGS-CD.

We now define formally the set-wise CD algorithms.

Definition 1 (*Set-wise CD algorithm*). In a set-wise CD algorithm, every coordinate $\ell \in [E]$ is assigned to (potentially multiple) sets $\mathcal{S}_i, i \in [n]$, such that all coordinates belong to at least one set. At any time, a set $\mathcal{S}_i$ may activate with uniform probability among the i; a set-wise CD algorithm then chooses a single coordinate $\ell \in \mathcal{S}_i$ to update using (4).

[2]The "+" stresses that $\gamma_{\min}^+$ is the smallest *strictly positive* eigenvalue.

The next remark shows how the decentralized problem (2) can be solved asynchronously with set-wise CD algorithms.

Remark 1. By letting (i) the E coordinates[3] in Definition 1 correspond to the dual variables $\lambda_\ell, \ell \in [E]$, and (ii) the $\mathcal{S}_i, i \in [n]$ be the sets of dual variables corresponding to the edges that are connected to each node i, nodes can run a set-wise CD algorithm to solve (3) (and thus, also (2)) asynchronously.

Furthermore, the set-wise CD algorithms can also be used in the parallel distributed setting, as explained below.

Remark 2. In a parallel distributed setting, where the parameter vector $x \in \mathbb{R}^E$ is stored[4] in a shared server and can be modified by n workers, the (primal) optimization can be done with set-wise CD algorithms by letting each worker $i \in [n]$ modify a subset $\mathcal{S}_i$ of coordinates such that when the worker activates it chooses one coordinate $\ell \in \mathcal{S}_i$ to modify, and all coordinates $\ell \in [E]$ belong to at least one set $\mathcal{S}_i, i \in [n]$.

We additionally note that, although in the parallel distributed setting each coordinate may belong to any number of sets between 1 and n, the results that we present here apply to the case where all coordinates belong to exactly two sets (i.e. they can be modified by exactly two workers), since our analysis is driven mainly by the decentralized setting.

In light of Remark 1, in the following we illustrate the steps that should be performed by the nodes to run the set-wise CD algorithms to find an optimal value λ^*. We first note that the gradient of F in the direction[5] of λ_ℓ for $\ell \equiv (i,j)$ is

$$\nabla_\ell F(\lambda) = A_{i\ell} \nabla f_i^*(u_i^T A\lambda) + A_{j\ell} \nabla f_j^*(u_j^T A\lambda). \tag{6}$$

The nodes can use (4) and (6) to update the λ_ℓ that they have access to (i.e., those corresponding to the edges they are connected to) as follows: each node i keeps in memory the current values of $\lambda_\ell, \ell \in \mathcal{S}_i$, which are needed to compute $\nabla f_i^*(u_i^T A\lambda)$. Then, when edge $\ell \equiv (i,j)$ is updated (either because node i activated and contacted j, or vice versa), both i and j compute their respective terms in the right-hand side of (6) and exchange them through their link. Finally, both nodes compute (6) and update their copy of λ_ℓ applying (4).

Algorithms 1 and 2 below detail these steps for SU-CD and SGS-CD, respectively. We have used $\mathcal{N}_i$ to indicate the set of neighbors of node i (note that $\mathcal{S}_i = \{\ell : \ell \equiv (i,j), j \in \mathcal{N}_i\}$). Table I shows this and other set-related notation that will be frequently used in the sections that follow.

We now proceed to describe the SU-CD and SGS-CD algorithms in detail, and prove their linear convergence rates.

TABLE I: Set-related definitions

$\mathcal{S}_i$	Set of edges connected to node i
$\mathcal{N}_i$	Set of neighbors of node i
N_i	Degree of node i, i.e. $N_i = \lvert\mathcal{S}_i\rvert = \lvert\mathcal{N}_i\rvert$
$N_{\max}$	Maximum degree in the network, i.e. $\max_i N_i$
T_i	Selector matrix of set $\mathcal{S}_i$ (see Definition 2)
$\mathcal{S}'_i$	Subset $\mathcal{S}'_i \subseteq \mathcal{S}_i$ such that $\mathcal{S}'_i \cap \mathcal{S}'_j = \emptyset$ if $i \neq j$
T'_i	Selector matrix of set $\mathcal{S}'_i$
$\overline{\mathcal{S}'_i}$	Complement set of $\mathcal{S}'_i$ such that $\overline{\mathcal{S}'_i} = \mathcal{S}_i \setminus \mathcal{S}'_i$
$\overline{T'_i}$	Selector matrix of set $\overline{\mathcal{S}'_i}$

A. Set-wise Uniform CD (SU-CD)

In SU-CD, the activated node chooses the neighbor uniformly at random, as shown in Alg. 1. We can compute the per-iteration progress of SU-CD taking expectation in (5):

$$\begin{aligned}\mathbb{E}\left[F(\lambda^{k+1}) \mid \lambda^k\right] &\leq F(\lambda^k) - \frac{1}{2L}\mathbb{E}\left[\left(\nabla_\ell F(\lambda^k)\right)^2 \mid \lambda^k\right] \\ &= F(\lambda^k) - \frac{1}{2Ln}\sum_{i=1}^n \frac{1}{N_i}\sum_{\ell\in\mathcal{S}_i}\left(\nabla_\ell F(\lambda^k)\right)^2 \\ &\leq F(\lambda^k) - \frac{1}{LnN_{\max}}\left\|\nabla F(\lambda^k)\right\|_2^2 \end{aligned} \tag{7}$$

where $N_i = |\mathcal{S}_i|$, $N_{\max} = \max_i N_i$, and the factor 2 in the denominator disappears because each coordinate $\ell \equiv (i,j)$ is counted twice (once in the sum through $\mathcal{S}_i$ and once in that through $\mathcal{S}_j$).

The standard procedure to show the linear convergence of CD in the single-machine case is to lower-bound $\|\nabla F(\lambda)\|_2^2$ using the strong convexity of the function [20], [21]. However, since F is *not* strongly convex (Lemma 1), we cannot apply this procedure to get the linear rate of SU-CD.

We can, however, use F's strong convexity in $\|\cdot\|_A$ instead (Lemma 2). The next result gives the core of the proof.

Lemma 3. It holds that

$$\|\nabla F(\lambda)\|_2 = \|\nabla F(\lambda)\|_A = \|\nabla F(\lambda)\|_A^*, \tag{8}$$

where $\|\cdot\|_A^*$ is the dual norm of $\|\cdot\|_A$, defined as (e.g. [29])

$$\|z\|_A^* = \sup_{x\in\mathbb{R}^d}\left\{z^T x \,\middle|\, \|x\|_A \leq 1\right\}. \tag{9}$$

Proof. Note that $\forall w \neq \mathbf{0}$ such that $F(\lambda + tw) = F(\lambda)\ \forall t$, it holds that $w^T\nabla F(\lambda) = 0$ and thus $\nabla F(\lambda) \in \text{range}(A^T)$. This means that $A^+A\nabla F(\lambda) = I_E\nabla F(\lambda)$, and therefore it holds that $\|\nabla F(\lambda)\|_A = \|\nabla F(\lambda)\|_2$. Finally, since the dual norm of the L2 norm is the L2 norm itself, we have that also $\|\nabla F(\lambda)\|_A^* = \|\nabla F(\lambda)\|_2$, which gives the result. □

We now use Lemma 3 to prove the linear rate of SU-CD.

Theorem 1 (***Rate of SU-CD***)**.** SU-CD converges as

$$\mathbb{E}\left[F(\lambda^{k+1}) \mid \lambda^k\right] - F(\lambda^*) \leq \left(1 - \frac{2\sigma_A}{LnN_{\max}}\right)\left[F(\lambda^k) - F(\lambda^*)\right].$$

Proof. Since $F(\lambda)$ is strongly convex in $\|\cdot\|_A$ with strong convexity constant σ_A (Lemma 2), it holds

$$F(y) \geq F(x) + \langle\nabla F(x), y - x\rangle + \frac{\sigma_A}{2}\|y - x\|_A^2.$$

Minimizing both sides with respect to y as in [21] we get

[3] If $d > 1$, the standard CD terminology calls each λ_ℓ a "block coordinate", i.e. a vector of d coordinates out of the $E \cdot d$ of $F : \mathbb{R}^{E\cdot d} \to \mathbb{R}$.

[4] Also in this case we may actually have $x \in \mathbb{R}^{d\cdot E}$ and each worker may update a block coordinate $x_\ell \in \mathbb{R}^d$ at each iteration.

[5] This is equivalent to saying "the ℓ-th (block) entry of the gradient ∇F".

Algorithm 1 Set-wise Uniform CD (SU-CD)

1: **Input:** Functions f_i, step η, incidence matrix A, graph $\mathcal{G}$
2: Initialize $\theta_i^0, i = 1, \dots, n$ and $\lambda_\ell^0, \ell = 1, \dots, E$
3: **for** $k = 1, 2, \dots$ **do**
4: Sample activated node $i \in \{1, \dots, n\}$ uniformly
5: Node i picks neighbor $j \leftarrow \mathcal{U}\{h : h \in \mathcal{N}_i\}$
6: Node i computes $\nabla f_i^*(u_i^T A\lambda)$ and sends it to j
7: Node j computes $\nabla f_j^*(u_j^T A\lambda)$ and sends it to i
8: Nodes i, j: $(i, j) \equiv \ell$ use (6) to update their local copies of λ_ℓ by $\lambda_\ell^k \leftarrow \lambda_\ell^{k-1} - \eta \nabla_\ell F(\lambda)$
9: $\lambda_m^k \leftarrow \lambda_m^{k-1}\ \forall$ edges $m \neq \ell$

Algorithm 2 Set-wise Gauss-Southwell CD (SGS-CD)

1: **Input:** Functions f_i, step η, incidence matrix A, graph $\mathcal{G}$
2: Initialize $\theta_i^0, i = 1, \dots, n$ and $\lambda_\ell^0, \ell = 1, \dots, E$
3: **for** $k = 1, 2, \dots$ **do**
4: Sample activated node $i \in \{1, \dots, n\}$ uniformly
5: All $h \in \mathcal{N}_i$ compute $\nabla f_h^*(u_h^T A\lambda)$ and send it to i
6: Node i computes $\nabla f_i^*(u_i^T A\lambda)$
7: Compute $\nabla_\ell F(\lambda)\ \forall \ell \in \mathcal{S}_i$ (equivalently, $\forall h \in \mathcal{N}_i$) with (6) using the received $\nabla f_h^*(u_h^T A\lambda)$
8: Node i selects $j \leftarrow \max_{h \in \mathcal{N}_i} |\nabla_\ell F(\lambda)|, \ell \equiv (i, h)$
9: Node i sends $\nabla f_i^*(u_i^T A\lambda)$ to j
10: Nodes i, j: $(i, j) \equiv \ell$ use (6) to update their local copies of λ_ℓ by $\lambda_\ell^k \leftarrow \lambda_\ell^{k-1} - \eta \nabla_\ell F(\lambda)$
11: $\lambda_m^k \leftarrow \lambda_m^{k-1}\ \forall$ edges $m \neq \ell$

$$F(x^*) \geq F(x) - \frac{1}{2\sigma_A} \left(\|\nabla F(x)\|_A^* \right)^2, \tag{10}$$

and rearranging terms we obtain the lower bound $\left(\|\nabla F(x)\|_A^* \right)^2 \geq 2\sigma_A (F(x) - F(x^*))$.

Finally, we can use Lemma 3 to replace $\|\nabla F(x)\|_2^2$ with $\left(\|\nabla F(x)\|_A^* \right)^2$ in (7), and use the lower bound on $\left(\|\nabla F(x)\|_A^* \right)^2$ to get the result. □

Note that vector λ has $\frac{1}{2}\sum_i N_i = E \leq \frac{nN_{\max}}{2}$ coordinates, where the inequality holds with equality for regular graphs. We make the following remark.

Remark 3. If $\mathcal{G}$ is regular, the linear convergence rate of SU-CD is $\frac{\sigma_A}{LE}$, which matches the rate of single-machine uniform CD for strongly convex functions [20], [21], with the only difference that now the strong convexity constant σ_A is defined over norm $\|\cdot\|_A$.

In the next section we analyze SGS-CD and show that its convergence rate can be up to $N_{\max}$ times that of SU-CD.

B. Set-wise Gauss-Southwell CD (SGS-CD)

In SGS-CD, as shown in Alg. 2, the activated node i selects the neighbor j to contact applying the GS rule within the edges in $\mathcal{S}_i$:

$$\ell = \underset{m \in \mathcal{S}_i}{\operatorname{argmax}} \left(\nabla_m F(\lambda) \right)^2,$$

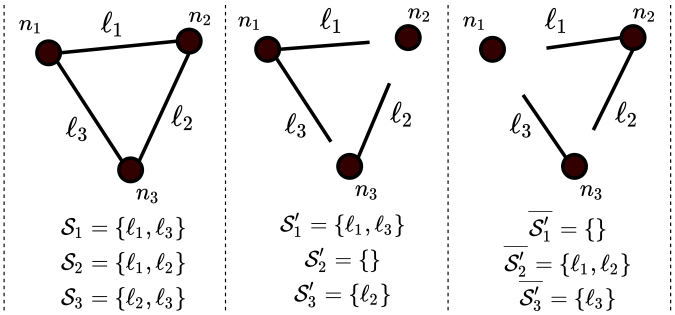


Fig. 1: Example of sets $\mathcal{S}_i$ and *one possibility* for $\mathcal{S}_i'$ and $\overline{\mathcal{S}_i'}$

and then j is the neighbor that satisfies $\ell \equiv (i, j)$. In order to make this choice, all nodes $h \in \mathcal{N}_i$ must send their $\nabla f_h^*(u_h^T A\lambda)$ to node i (line 5 in Alg. 2). We discuss this additional communication step of SGS-CD with respect to SU-CD in Sec. VIII.

To obtain the convergence rate of SGS-CD we will follow the steps taken for SU-CD in the proof of Theorem 1. As done for SU-CD, we start by computing the per-iteration progress taking expectation in (5) for SGS-CD:

$$\mathbb{E}\left[F(\lambda^{k+1}) \mid \lambda^k \right] \leq F(\lambda^k) - \frac{1}{2Ln} \sum_{i=1}^{n} \max_{\ell \in \mathcal{S}_i} \left(\nabla_\ell F(\lambda^k) \right)^2. \tag{11}$$

Given this per-iteration progress, to proceed as we did for SU-CD we need to show (i) that the sum on the right-hand side of (11) defines a norm, and (ii) that strong convexity holds in its dual norm. We start by defining the selector matrices T_i, which will significantly simplify notation.

Definition 2 (***Selector matrices***)**.** The selector matrices $T_i \in \{0, 1\}^{N_i \times E}, i = 1, \dots, n$ select the coordinates of a vector in $\mathbb{R}^E$ that belong to set $\mathcal{S}_i$. Note that any vertical stack of the unitary vectors $\left\{ e_\ell^T \right\}_{\ell \in \mathcal{S}_i}$ gives a valid T_i.

We can now show that the sum in (11) is a (squared) norm. Since the operation involves applying $\max(\cdot)$ within each set $\mathcal{S}_i$, we will denote this norm $\|x\|_{\text{SM}}$, where the subscript SM stands for "Set-Max".

Lemma 4. The function $\|x\|_{\text{SM}} \triangleq \sqrt{\sum_{i=1}^n \|T_i x\|_\infty^2} = \sqrt{\sum_{i=1}^n \max_{j \in \mathcal{S}_i} x_j^2}$ is a norm in $\mathbb{R}^E$.

Proof. Using $\max_{j \in \mathcal{S}_i} \left(x_j^2 + y_j^2 \right) \leq \max_{j \in \mathcal{S}_i} x_j^2 + \max_{j \in \mathcal{S}_i} y_j^2$ and $\sqrt{a+b} \leq \sqrt{a} + \sqrt{b}$ we can show that $\|\cdot\|_{\text{SM}}$ satisfies the triangle inequality. It is straightforward to show that $\|\alpha x\|_{\text{SM}} = |\alpha| \, \|x\|_{\text{SM}}$ and $\|x\|_{\text{SM}} = 0$ if and only if $x = \mathbf{0}$. □

Following the proof of Theorem 1, we would like to show that F is strongly convex in the dual norm $\|\cdot\|_{\text{SM}}^*$. Furthermore, we would like to compare the strong convexity constant σ_{SM} with σ_A to quantify the speedup of SGS-CD with respect to SU-CD. It turns out, though, that computing $\|\cdot\|_{\text{SM}}^*$ is not easy at all; the main difficulty stems from the fact that the sets $\mathcal{S}_i$ are overlapping (or non-disjoint), since each coordinate $\ell \equiv (i, j)$ belongs to both $\mathcal{S}_i$ and $\mathcal{S}_j$. The first scheme in Figure 1 illustrates this fact for the 3-node clique.

To circumvent this issue, we define a new norm $\|\cdot\|_{\text{SMNO}}^*$ ("Set-Max Non-Overlapping") that we can directly relate to

$\|\cdot\|^*_{\text{SM}}$ (Lemma 5) and whose value we can compute explicitly (Lemma 6), which will later allow us to relate the three strong convexity constants $\sigma_{\text{SM}}, \sigma_{\text{SMNO}}$, and σ_A (Theorem 2).

Definition 3 (***Norm*** $\|\cdot\|^*_{\textit{SMNO}}$)**.** We assume that each coordinate $\ell \equiv (i,j)$ is assigned to only one of the sets $\mathcal{S}'_i \subseteq \mathcal{S}_i$ or $\mathcal{S}'_j \subseteq \mathcal{S}_j$, such that the new sets $\{\mathcal{S}'_i\}_{i=1}^n$ are non-overlapping (some sets can be empty), and all coordinates ℓ belong to *exactly one* set in $\{\mathcal{S}'_i\}$. We name the selector matrices of these new sets T'_i, so that each possible choice of $\{\mathcal{S}'_i\}$ defines a different set $\{T'_i\}$. Then, we define

$$\|z\|^*_{\text{SMNO}} = \sup_x \left\{ z^T x \;\middle|\; \sqrt{\sum_{i=1}^n \|T_i^* x\|_\infty^2} \leq 1 \right\}, \tag{12}$$

with the choice of non-overlapping sets

$$\{T_i^*\} = \arg\max_{\{T'_i\}} \sum_{i=1}^n \|T'_i x\|_\infty^2 . \tag{13}$$

Note that the maximizations in (12) and (13) are coupled. We denote the value of x that attains (12) by x^*_{SMNO}.

The definition of sets $\mathcal{S}'_i$ corresponds to assigning each edge ℓ to one of the two nodes at its endpoints, as illustrated in the second scheme of Figure 1. Therefore, for each possible pair $(\{\mathcal{S}'_h\}, \{\underline{T}'_h\}), h \in [n]$ we can define a complementary pair $(\{\mathcal{S}'_h\}, \{\overline{T}'_h\})$ such that if $\ell \equiv (i,j)$ was assigned to $\mathcal{S}'_i$ in $\{\mathcal{S}'_h\}$, then it is assigned to $\overline{\mathcal{S}'_j}$ in $\{\overline{\mathcal{S}'_h}\}$. This corresponds to assigning ℓ to the opposite endpoint (node) to the one originally chosen, as shown in the third scheme of Figure 1. With these definitions, it holds (potentially with some permutation of the rows):

$$T_i = \begin{bmatrix} T'_i \\ \overline{T}'_i \end{bmatrix} = \begin{bmatrix} T'_i \\ \mathbf{0} \end{bmatrix} + \begin{bmatrix} \mathbf{0} \\ \overline{T}'_i \end{bmatrix}, \; i = 1, \ldots, n.$$

We remark that the equality above holds for *any* $\{T'_i\}$ corresponding to a feasible assignment $\{\mathcal{S}'_i\}$, and in particular it hols for $(\{\mathcal{S}_i^*\}, \{T_i^*\})$. This fact is used in the proof of the following lemma, which relates norms $\|\cdot\|^*_{\text{SM}}$ and $\|\cdot\|^*_{\text{SMNO}}$. This will allow us to complete the analysis with $\|\cdot\|^*_{\text{SMNO}}$, which we can compute explicitly (Lemma 6).

Lemma 5. The dual norm of $\|\cdot\|_{\text{SM}}$, denoted $\|\cdot\|^*_{\text{SM}}$, satisfies $\frac{1}{2}\left(\|z\|^*_{\text{SMNO}}\right)^2 \leq \left(\|z\|^*_{\text{SM}}\right)^2 \leq \left(\|z\|^*_{\text{SMNO}}\right)^2$.

Proof. By definition

$$\|z\|^*_{\text{SM}} = \sup_x \left\{ z^T x \;\middle|\; \sqrt{\sum_{i=1}^n \|T_i x\|_\infty^2} \leq 1 \right\}. \tag{14}$$

By inspection we can tell that the x that attains the supremum, denoted x^*_{SM}, will satisfy $\sum_{i=1}^n \|T_i x^*_{\text{SM}}\|_\infty^2 = 1$. Similarly, x^*_{SMNO} (defined under (13)) must satisfy $\sum_{i=1}^n \|T_i^* x^*_{\text{SMNO}}\|_\infty^2 = 1$. Note that in these two equalities the $\{T_i\}$ are overlapping sets and the $\{T_i^*\}$ are non-overlapping. Therefore, in order to satisfy both equalities it must hold that $|[x^*_{\text{SM}}]_\ell| \leq |[x^*_{\text{SMNO}}]_\ell|, \ell \in [E]$, i.e. the magnitude of the entries of x^*_{SMNO} are equal or larger than the magnitudes of the corresponding entries of x^*_{SM}. Referring to the definitions (12) and (14), this means that $\|z\|^*_{\text{SM}} \leq \|z\|^*_{\text{SMNO}}$.

We now proceed to show the first inequality in the lemma. We note that

$$\begin{aligned} \sum_{i=1}^n \|T_i x\|_\infty^2 &= \sum_{i=1}^n \left\| \begin{bmatrix} T'_i \\ \mathbf{0} \end{bmatrix} x + \begin{bmatrix} \mathbf{0} \\ \overline{T}'_i \end{bmatrix} x \right\|_\infty^2 \\ &\leq \sum_{i=1}^n \|T'_i x\|_\infty^2 + \sum_{i=1}^n \left\|\overline{T}'_i x\right\|_\infty^2 \leq 2 \sum_{i=1}^n \left\|\widehat{T}'_i x\right\|_\infty^2, \end{aligned} \tag{15}$$

with

$$\{\widehat{T}'_i\} = \arg \max_{\{T'_i\},\{\overline{T}'_i\}} \left(\sum_{i=1}^n \|T'_i x\|_\infty^2 , \sum_{i=1}^n \left\|\overline{T}'_i x\right\|_\infty^2 \right). \tag{16}$$

We now evaluate (15) and (16) at x^*_{SMNO}. Due to (13) we have $\{\widehat{T}'_i\} = \{T_i^*\}$, and since $\sum_{i=1}^n \|T_i^* x^*_{\text{SMNO}}\|_\infty^2 = 1$, the rightmost member of (15) takes value 2. Then, dividing both sides of (15) by 2 we obtain

$$\frac{1}{2}\sum_{i=1}^n \|T_i x^*_{\text{SMNO}}\|_\infty^2 = \sum_{i=1}^n \left\| T_i \frac{x^*_{\text{SMNO}}}{\sqrt{2}} \right\|_\infty^2 \leq 1,$$

and since $\sum_{i=1}^n \|T_i x^*_{\text{SM}}\|_\infty^2 = 1$, we conclude that it must hold that $\frac{1}{\sqrt{2}}|[x^*_{\text{SMNO}}]_\ell| \leq |[x^*_{\text{SM}}]_\ell|, \ell \in [E]$, and thus $\frac{1}{\sqrt{2}}\|z\|^*_{\text{SMNO}} \leq \|z\|^*_{\text{SM}}$. □

The next lemma gives the value of $\|x\|^*_{\text{SMNO}}$ explicitly, which will be needed to compare the strong convexity constant σ_{SMNO} with σ_A.

Lemma 6. It holds that $\|x\|^*_{\text{SMNO}} = \sqrt{\sum_{i=1}^n \|T_i^* x\|_1^2}$.

Proof. Since the sets $\{\mathcal{S}_i^*\}$ are non-overlapping and in (12) norm $\|\cdot\|_\infty$ is applied per-set, the entries x_ℓ of x^*_{SMNO} will have $|x_\ell| = x^{(i)} \geq 0 \; \forall \, \ell \in \mathcal{S}_i^*$ and the sign will match that of the entries of z, i.e. $\text{sign}(x_\ell) = \text{sign}(z_\ell)$. The maximization of (12) then becomes

$$\begin{aligned} \underset{\{x^{(i)}\}}{\text{maximize}} \quad & \sum_{i=1}^n \sum_{\ell \in \mathcal{S}_i^*} \left(|z_\ell| \cdot x^{(i)} \right) \\ \text{subject to} \quad & \sqrt{\sum_{i=1}^n \left(x^{(i)}\right)^2} \leq 1. \end{aligned}$$

Factoring out $x^{(i)}$ in the objective and noting that $\sum_{\ell \in \mathcal{S}_i^*} |z_\ell| = \|T_i^* z\|_1$, we can define $w = [x^{(1)}, \ldots, x^{(n)}]^T$ and $y = \left[\|T_1^* z\|_1, \ldots, \|T_n^* z\|_1\right]^T$ so that (12) now reads

$$\|z\|^*_{\text{SMNO}} = \sup_w \left\{ y^T w \;\middle|\; \|w\|_2 \leq 1 \right\}.$$

The right-hand side is the definition of $\|\cdot\|_2^*$, the dual of the L2 norm, evaluated at y. Since $\|\cdot\|_2^* = \|\cdot\|_2$, we have that $\|z\|^*_{\text{SMNO}} = \|y\|_2 = \sqrt{\sum_{i=1}^n \|T_i^* z\|_1^2}$. □

We can now prove the linear convergence rate of SGS-CD.

Theorem 2 (***Rate of SGS-CD***)**.** SGS-CD converges as

$$\mathbb{E}\left[F(\lambda^{k+1}) \mid \lambda^k\right] - F(\lambda^*) \leq \left(1 - \frac{\sigma_{\text{SM}}}{Ln}\right)\left[F(\lambda^k) - F(\lambda^*)\right],$$

with

$$\frac{\sigma_A}{N_{\max}} \leq \sigma_{\text{SM}} \leq 2\sigma_A. \tag{17}$$

Proof. Similarly to what we did for SU-CD, we can depart from the strong convexity of F in the $\|\cdot\|_{\text{SM}}$ norm:

$$F(y) \geq F(x) + \langle \nabla F(x), y - x\rangle + \frac{\sigma_{\text{SM}}}{2}\left(\|y-x\|^*_{\text{SM}}\right)^2,$$

then minimize both sides with respect to y to obtain

$$F(x^*) \geq F(x) - \frac{1}{2\sigma_{\text{SM}}}\left(\|\nabla F(x)\|_{\text{SM}}\right)^2, \tag{18}$$

which is analogous to (10), and then rearrange terms to obtain a lower bound on $\|\nabla F(\lambda)\|^2_{\text{SM}}$. Using this lower bound in (11) gives the rate of SGS-CD.

Since this rate is given in terms of σ_{SM} and that of SU-CD in Theorem 1 is given in terms of σ_A, we need (17) to compare both rates. However, we cannot prove these inequalities directly because we cannot compare norms $\|\cdot\|_A$ and $\|\cdot\|^*_{\text{SM}}$ (due to the overlap of the coordinate sets, which prevents us from computing the latter). However, we *can* compare $\|\cdot\|_A$ with $\|\cdot\|^*_{\text{SMNO}}$ and $\|\cdot\|^*_{\text{SM}}$ with $\|\cdot\|^*_{\text{SMNO}}$ individually, from which we will obtain (17). In particular, we will show the inequalities

$$\frac{\sigma_A}{N_{\max}} \leq \sigma_{\text{SMNO}} \leq \sigma_A \tag{19}$$

and

$$\sigma_{\text{SMNO}} \leq \sigma_{\text{SM}} \leq 2\sigma_{\text{SMNO}}. \tag{20}$$

We start by proving (19). Below we assume $x \in \text{range}(A^T)$; the results here can then be directly applied to the proofs above because $\|\cdot\|_A, \|\cdot\|_{\text{SM}}, \|\cdot\|_{\text{SMNO}}$ and their duals are applied to ∇F, which is always in $\text{range}(A^T)$ (Lemma 3).

For $x \in \text{range}(A^T)$ it holds that (Lemmas 3 and 6):

$$\|x\|^2_A = \|x\|^2_2 = \sum_{i=1}^{E} x_i^2 = \sum_{i=1}^{n} \|T_i^* x\|_2^2$$
$$\left(\|x\|^*_{\text{SMNO}}\right)^2 = \sum_{i=1}^{n} \|T_i^* x\|_1^2.$$

We also note that, using the Cauchy-Schwarz inequality and denoting $[v]_i$ the i^{th} entry of vector v, it holds both that

$$\sum_{i=1}^{n} \|T_i^* x\|_2^2 \leq \sum_{i=1}^{n}\left(\sum_{j\in\mathcal{S}_i^*} |x_j|\right)^2 = \sum_{i=1}^{n}\|T_i^* x\|_1^2, \text{ and}$$
$$\sum_{i=1}^{n} \|T_i^* x\|_1^2 = \sum_{i=1}^{n}\left(\mathbf{1}^T\left[\left|[T_i^* x]_1\right|, \ldots, \left|[T_i^* x]_{N_i^*}\right|\right]^T\right)^2$$
$$\overset{\text{C.S.}}{\leq} \sum_{i=1}^{n} N_i^* \|T_i^* x\|_2^2 \leq N_{\max}\sum_{i=1}^{n}\|T_i^* x\|_2^2,$$

where $N_i^* = |\mathcal{S}_i^*|$. We can summarize these relations as

$$\frac{1}{N_{\max}}\left(\|x\|^*_{\text{SMNO}}\right)^2 \leq \|x\|^2_A \leq \left(\|x\|^*_{\text{SMNO}}\right)^2.$$

Using these inequalities in the strong convexity definitions, similarly to [21], we get both

$$\begin{aligned} F(y) &\geq F(x) + \langle \nabla F(x), y-x\rangle + \frac{\sigma_A}{2}\left(\|y-x\|_A\right)^2 \\ &\geq F(x) + \langle \nabla F(x), y-x\rangle + \frac{\sigma_A}{2N_{\max}}\left(\|y-x\|^*_{\text{SMNO}}\right)^2, \end{aligned} \tag{21}$$

and

$$\begin{aligned} F(y) &\geq F(x) + \langle \nabla F(x), y-x\rangle + \frac{\sigma_{\text{SMNO}}}{2}\left(\|y-x\|^*_{\text{SMNO}}\right)^2 \\ &\geq F(x) + \langle \nabla F(x), y-x\rangle + \frac{\sigma_{\text{SMNO}}}{2}\left(\|y-x\|_A\right)^2. \end{aligned} \tag{22}$$

Equation (21) says that F is at least $\frac{\sigma_A}{N_{\max}}$-strongly convex in $\|\cdot\|^*_{\text{SMNO}}$, and eq. (22) says that F is at least σ_{SMNO}-strongly convex in $\|\cdot\|_A$. Together they imply (19).

We can show (20) by the same procedure applied in eqs. (21) and (22), but now using the strong convexity of F in norms $\|\cdot\|^*_{\text{SM}}$ and $\|\cdot\|^*_{\text{SMNO}}$ together with Lemma 5. From $\frac{1}{2}(\|z\|^*_{\text{SMNO}})^2 \leq (\|z\|^*_{\text{SM}})^2$ we get $\frac{1}{2}\sigma_{\text{SM}} \leq \sigma_{\text{SMNO}}$, and from $(\|z\|^*_{\text{SM}})^2 \leq (\|z\|^*_{\text{SMNO}})^2$ we get $\sigma_{\text{SMNO}} \leq \sigma_{\text{SM}}$.

Finally, putting (19) and (20) together gives (17). □

Theorems 1 and 2 together allow us to compare the convergence rates of SU-CD and SGS-CD. We note that when σ_{SM} takes the upper value in (17), SGS-CD is (in expectation) $N_{\max}$ times faster than SU-CD. The lower bound in (17), on the other hand, suggests that SGS-CD could be slower than SU-CD. We remark that (in expectation) this is not true and the lower bound is vacuous, since the following holds.

Remark 4. For the same sequence of node activations, the suboptimality reduction of SGS-CD at each iteration is equal to or larger than that of SU-CD.

Taking this fact into account, we have the following result.

Corollary 3. In expectation, SGS-CD converges at least as fast as SU-CD, and can be up to $N_{\max}$ times faster.

Note that this result is analogous to that of [21] for single-machine CD, where they show that the GS rule can be up to d times faster than uniform sampling, d being the dimensionality of the problem.

We remark that achieving the upper bound of $N_{\max}$ speedup may require designing a scenario particularly favorable to SGS-CD with respect to SU-CD. Similarly, finding a setting where the former converges at the same speed as the latter also requires designing a particularly adversarial setting.

In our simulations of Section VII for the decentralized setting, SGS-CD achieves a speedup approximately in the middle of the range between 1 and $N_{\max}$. We show that this speedup *increases linearly with* $N_{\max}$, achieving remarkable gains in terms of suboptimality reduction versus number of iterations (see Fig. 2). Furthermore, in the same figure we show that for the parallel distributed setting *the maximum speedup of* $N_{\max}$ *is attainable*. We explain this further in Section VI-A.

V. Set-wise Lipschitz CD Algorithms

In Section IV we stated that the dual function F is L-smooth and therefore a sufficient condition for the set-wise algorithms to converge was using stepsize $\eta = 1/L$. However, the updates of some (and maybe *many*) coordinates could use larger stepsizes by exploiting the fact F has coordinate-wise smoothness $L_\ell \leq L$, i.e. for $\alpha \in \mathbb{R}$:

$$|\nabla_\ell F(\lambda + \alpha e_\ell) - \nabla_\ell F(\lambda)| \leq L_\ell \alpha. \tag{23}$$

Therefore, when the coordinate-wise Lipschitz constants L_ℓ are known or can be estimated (see Section V-C) we can apply the update (4) with stepsize $\eta^k = 1/L_\ell$, with ℓ being the coordinate updated at iteration k.

In the sections that follow we show that by using the knowledge (or estimation) of the coordinate-wise Lipschitz constants and per-coordinate stepsizes we can have:

1) An algorithm that has randomized but non-uniform neighbor selection that is provably faster than SU-CD. We call this algorithm Set-wise Lipschitz CD (SL-CD).
2) An algorithm that applies locally the Gauss-Southwell Lipschitz rule [21] and that converges provably faster than both SL-CD and SGS-CD. We call this algorithm Set-wise GS Lipschitz CD (SGSL-CD).

Once again, while the seminal work of [21] has analyzed both of these rules in the context of single-machine coordinate descent, their adaptation to set-wise CD brings important new challenges. In this section, we prove that SL-CD is at least as fast as SU-CD, and that SGSL-CD is at least as fast as the fastest algorithm between SGS-CD and SL-CD.

In the proofs that follow we will use the following fact.

Fact 1. Denote $a \circ b$ the per-entry product of vectors a and b. Then, for any norm $\|\cdot\|$ and finite $a : a_i > 0 \ \forall i$, if we define $\|x\|_a := \|a \circ x\|$, then $\|x\|_a^* := \|a_{-1} \circ x\|^*$ with $a_{-1} = [\frac{1}{a_1}, \dots, \frac{1}{a_d}]$.

Proof. By definition

$$\|z\|_a^* = \sup_{\|x\|_a \le 1} z^T x,$$

and defining $y := a \circ x$ we get

$$\|z\|_a^* = \sup_{\|y\| \le 1} z^T (a_{-1} \circ y) = \|a_{-1} \circ z\|^* .$$

□

A. Set-wise Lipschitz CD (SL-CD)

In SL-CD, an activated node i chooses the edge $\ell \in \mathcal{S}_i$ to update at random with probability

$$p_\ell = \frac{L_\ell}{\sum_{m \in \mathcal{S}_i} L_m} \tag{24}$$

and updates λ_ℓ applying (4) with stepsize $\eta^k = 1/L_\ell$.

For convenience, we define the quantities

$$L^{(i)} := \sum_{m \in \mathcal{S}_i} L_m$$

and

$$\mathcal{L}_\ell := \left(\frac{1}{L^{(i)}} + \frac{1}{L^{(j)}} \right) \quad \text{for} \quad \ell \equiv (i,j).$$

With these definitions, and taking expectation in (5) for the Lipschitz-dependent sampling probabilities (24) gives

$$\begin{aligned}
\mathbb{E}[F(\lambda^{k+1}) \mid \lambda^k] &\le F(\lambda^k) - \frac{1}{2} \mathbb{E}\left[\frac{1}{L_{\ell_k}} \left[\nabla_{\ell_k} F(\lambda^k) \right]^2 \right] \\
&= F(\lambda^k) - \frac{1}{2n} \sum_{i=1}^n \frac{1}{L^{(i)}} \sum_{\ell \in \mathcal{S}_i} \left[\nabla_\ell F(\lambda^k) \right]^2 \\
&\overset{(a)}{=} F(\lambda^k) - \frac{1}{2n} \sum_{\ell=1}^E \mathcal{L}_\ell \left[\nabla_\ell F(\lambda^k) \right]^2
\end{aligned}$$

where in (a) we used that $\ell \equiv (i,j)$ implies $\ell \in \mathcal{S}_i, \mathcal{S}_j$.

In order to prove the convergence rate of SL-CD, provided in Theorem 4, we define the norm

$$\|x\|_{\mathcal{L}} := \sqrt{\sum_{\ell=1}^E \mathcal{L}_\ell x_\ell^2},$$

so that we can write the per-iteration progress of SL-CD as

$$\mathbb{E}[F(\lambda^{k+1})] \le F(\lambda^k) - \frac{1}{2n} \left\| \nabla F(\lambda^k) \right\|_{\mathcal{L}}^2 . \tag{25}$$

Noting that $\|x\|_{\mathcal{L}} = \left\| x \circ \left[\sqrt{\mathcal{L}_1}, \dots, \sqrt{\mathcal{L}_E} \right] \right\|_2$ we can apply Fact 1 to get its dual norm:

$$\|x\|_{\mathcal{L}}^* = \sqrt{\sum_{\ell=1}^E \frac{1}{\mathcal{L}_\ell} x_\ell^2}.$$

We call $\sigma_{\mathcal{L}}$ the strong convexity constant of F in this norm:

$$F(y) \ge F(x) + \langle \nabla F(x), y - x \rangle + \frac{\sigma_{\mathcal{L}}}{2} (\|y - x\|_{\mathcal{L}}^*)^2. \tag{26}$$

We use the definitions of $\|\cdot\|_{\mathcal{L}}^*$ and $\sigma_{\mathcal{L}}$ in the proof of the linear rate of SL-CD, given in the theorem below.

Theorem 4 (***Rate of SL-CD***)**.** SL-CD converges as

$$\mathbb{E}\left[F(\lambda^{k+1}) \mid \lambda^k \right] - F(\lambda^*) \le \left(1 - \frac{\sigma_{\mathcal{L}}}{n} \right) \left[F(\lambda^k) - F(\lambda^*) \right]$$

and it holds that

$$\sigma_A \mathcal{L}_{\min} \le \sigma_{\mathcal{L}} \le \sigma_A \mathcal{L}_{\max} \tag{27}$$

with $\mathcal{L}_{\min} = \min_\ell \mathcal{L}_\ell$ and $\mathcal{L}_{\max} = \max_\ell \mathcal{L}_\ell$.

Proof. We start by proving the linear rate. Minimizing both sides of (26) with respect to y as done in (10) and (18) we get

$$F(x^*) \ge F(x) - \frac{1}{2\sigma_{\mathcal{L}}} \|\nabla F(x)\|_{\mathcal{L}}^2 .$$

Rearranging terms gives a lower bound on $\|\nabla F(x)\|_{\mathcal{L}}^2$, and replacing in (25) gives the result.

We now move on to show (27). Once again, since the norms are evaluated at $\nabla F(\lambda)$ and (8) holds, to obtain the relation between $\sigma_{\mathcal{L}}$ and σ_A we will compare $\|\cdot\|_{\mathcal{L}}^*$ with $\|\cdot\|_2$ directly. We have that

$$c \|x\|_2^2 - (\|x\|_{\mathcal{L}}^*)^2 = c \sum_\ell x_\ell^2 - \sum_\ell \frac{1}{\mathcal{L}_\ell} x_\ell^2 = \sum_\ell \left(c - \frac{1}{\mathcal{L}_\ell} \right) x_\ell^2.$$

For $c \ge \max_\ell \frac{1}{\mathcal{L}_\ell} = \frac{1}{\mathcal{L}_{\min}}$ the expression is larger than zero, and thus

$$\frac{1}{\mathcal{L}_{\min}} \|x\|_2^2 \ge (\|x\|_{\mathcal{L}}^*)^2. \tag{28}$$

Similarly, we have that

$$c \|x\|_{\mathcal{L}}^2 - \|x\|_2^2 = \sum_\ell \left(\frac{c}{\mathcal{L}_\ell} - 1 \right) x_\ell^2$$

is larger than zero for $c \ge \mathcal{L}_{\max}$, and therefore

$$\mathcal{L}_{\max} (\|x\|_{\mathcal{L}}^*)^2 \ge \|x\|_2^2 . \tag{29}$$

Using these inequalities (and Lemma 3) in the strong convexity definitions we have on the one hand:

$$f(y) \geq f(x) + \langle \nabla f(x), y - x \rangle + \frac{\sigma_A}{2} \|y - x\|_A^2 \geq f(x) + \langle \nabla f(x), y - x \rangle + \frac{\sigma_A \mathcal{L}_{\min}}{2} (\|y - x\|_{\mathcal{L}}^*)^2, \tag{30}$$

and on the other hand:

$$f(y) \geq f(x) + \langle \nabla f(x), y - x \rangle + \frac{\sigma_{\mathcal{L}}}{2} (\|y - x\|_{\mathcal{L}}^*)^2 \geq f(x) + \langle \nabla f(x), y - x \rangle + \frac{\sigma_{\mathcal{L}}}{2\mathcal{L}_{\max}} \|y - x\|_A^2 . \tag{31}$$

Eqs. (30) and (31) indicate respectively that $\sigma_{\mathcal{L}} \geq \sigma_A \mathcal{L}_{\min}$ and that $\sigma_A \geq \frac{\sigma_{\mathcal{L}}}{\mathcal{L}_{\max}}$. Putting both together gives (27). □

Having obtained the rate of SL-CD, we can compare it against that of SU-CD. We have the following result.

Corollary 5. In expectation, SL-CD converges as fast or faster than SU-CD.

Proof. The convergence rate of SU-CD is $\frac{2\sigma_A}{LnN_{\max}}$ (Theorem 1) and that of SL-CD is $\frac{\sigma_{\mathcal{L}}}{n}$ (Theorem 4). Since in the slowest case of SL-CD we have $\sigma_{\mathcal{L}} = \sigma_A \mathcal{L}_{\min}$, it suffices to show that $\mathcal{L}_{\min} \geq \frac{2}{LN_{\max}}$. Indeed, we have that

$$L^{(i)} = \sum_{\ell \in \mathcal{S}_i} L_\ell \leq L_{\max} |\mathcal{S}_i| \leq L_{\max} N_{\max}$$

and therefore

$$\mathcal{L}_{\min} = \min_{(i,j) \in \mathcal{E}} \left(\frac{1}{L^{(i)}} + \frac{1}{L^{(j)}} \right) \geq \frac{2}{L_{\max} N_{\max}}$$

The proof is complete by noting that it always holds that $L_{\max} \leq L$ [30]. □

Since both SL-CD and SGS-CD can converge at the same speed as SU-CD in the worst case, we cannot claim that either of them is faster than the other. We can, however, exploit the knowledge of the Lipschitz constants to get an improved version of the GS rule, known as the Gauss-Southwell Lipschitz rule [21], that when combined with per-coordinate stepsizes allows for faster convergence than both SGS-CD *and* SL-CD. We call this algorithm SGSL-CD, and we analyze it next.

B. Set-wise Gauss-Southwell Lipschitz CD (SGSL-CD)

If node i goes active, the Gauss-Southwell Lipschitz (GSL) rule chooses to update $\lambda_\ell, \ell \in \mathcal{S}_i$ according to

$$\ell = \operatorname*{argmax}_{m \in \mathcal{S}_i} \frac{|\nabla_m f(x^k)|}{\sqrt{L_m}}.$$

If we now use the GSL rule with the per-coordinate stepsizes $\eta^k = 1/L_\ell$, the per-iteration progress given by (5) becomes:

$$\mathbb{E}\left[F(\lambda^{k+1})\right] \leq F(\lambda^k) - \frac{1}{2n} \sum_{i=1}^n \max_{\ell \in \mathcal{S}_i} \left(\frac{1}{L_\ell} \left[\nabla_\ell F(\lambda^k)\right]^2 \right). \tag{32}$$

Note the resemblance of this expression with the per-iteration progress of SGS-CD in (11). Similarly to the previous procedures, we define the "Set-Max Lipschitz" norm:

$$\|x\|_{\text{SML}} := \sqrt{\sum_{i=1}^n \max_{\ell \in \mathcal{S}_i} \left(\frac{1}{L_\ell} x_\ell^2 \right)}, \tag{33}$$

and call σ_{SML} the strong convexity constant of F in the dual norm $\|\cdot\|_{\text{SML}}^*$

$$F(y) \geq F(x) + \langle \nabla F(x), y - x \rangle + \frac{\sigma_{\text{SML}}}{2} \left(\|y - x\|_{\text{SML}}^* \right)^2 . \tag{34}$$

We can now state the convergence rate of SGSL-CD.

Theorem 6 (***Rate of SGSL-CD***)**.** SGSL-CD converges as

$$\mathbb{E}\left[F(\lambda^{k+1}) \mid \lambda^k\right] - F(\lambda^*) \leq \left(1 - \frac{\sigma_{\text{SML}}}{n}\right) \left[F(\lambda^k) - F(\lambda^*)\right], \tag{35}$$

and it holds that

$$\frac{\sigma_{\text{SM}}}{L} \leq \sigma_{\text{SML}}. \tag{36}$$

Proof. The expression of the rate is obtained with the procedure followed for the previous algorithms: minimizing both sides of (34) with respect to y and arranging terms we can find $\|\nabla F(x)\|_{\text{SML}} \geq 2\sigma_{\text{SML}}(F(x) - F(x^*))$, and using this in (32) gives (35).

We now show (36). By definition, the dual norm of $\|\cdot\|_{\text{SML}}$ is

$$\|z\|_{\text{SML}}^* := \sup_x \left\{ z^T x \;\middle|\; \|x\|_{\text{SML}} \leq 1 \right\} = \sup_x \left\{ z^T x \;\middle|\; \sqrt{\sum_{i=1}^n \max_{\ell \in \mathcal{S}_i} \left(\frac{1}{L_\ell} x_\ell^2 \right)} \leq 1 \right\}.$$

Similarly, we can write the dual norm of $\|\cdot\|_{\text{SM}}$ provided in (14) also as

$$\|z\|_{\text{SM}}^* := \sup_x \left\{ z^T x \;\middle|\; \|x\|_{\text{SM}} \leq 1 \right\} = \sup_x \left\{ z^T x \;\middle|\; \sqrt{\sum_{i=1}^n \max_{\ell \in \mathcal{S}_i} x_\ell^2} \leq 1 \right\}.$$

We call the values that achieve the supremum x_{SML}^* and x_{SM}^*, respectively. To maximize $z^T x$, these values will satisfy the constraints of each dual norm with equality, i.e.

$$\|x_{\text{SML}}^*\|_{\text{SML}} = 1 \quad \text{and} \quad \|x_{\text{SM}}^*\|_{\text{SM}} = 1.$$

From these conditions and the definitions of the dual norms above we obtain

$$x_{\text{SML}}^* \circ \left[\frac{1}{\sqrt{L_1}}, \cdots, \frac{1}{\sqrt{L_E}} \right] = x_{\text{SM}}^*.$$

Furthermore, using again $L_{\max} \leq L$, we have

$$x_{\text{SM}}^* = x_{\text{SML}}^* \circ \left[\frac{1}{\sqrt{L_1}}, \cdots, \frac{1}{\sqrt{L_E}} \right] \succeq \frac{1}{\sqrt{L_{\max}}} x_{\text{SML}}^* \succeq \frac{1}{\sqrt{L}} x_{\text{SML}}^*,$$

where "$\succeq$" indicates coordinate-wise inequality, and therefore

$$\|z\|_{\text{SM}}^* \geq \frac{1}{\sqrt{L}} \|z\|_{\text{SML}}^* .$$

Lastly, using this inequality in the strong convexity equation of F in $\|\cdot\|^*_{\text{SM}}$:

$$F(y) \geq F(x) + \langle \nabla F(x), y - x \rangle + \frac{\sigma_{\text{SM}}}{2} \left(\|y - x\|^*_{\text{SM}} \right)^2$$
$$\geq F(x) + \langle \nabla F(x), y - x \rangle + \frac{\sigma_{\text{SM}}}{2L} \left(\|y - x\|^*_{\text{SML}} \right)^2,$$

from where we obtain $\frac{\sigma_{\text{SM}}}{L} \leq \sigma_{\text{SML}}$ □

Theorem 6 states that SGSL-CD converges (in expectation) at least as fast as SGS-CD. Algorithm SGSL-CD is also at least as fast as SL-CD by an argument analogous to Remark 4: for the same sequence of node activations, the set-wise GSL rule achieves an equal or larger suboptimality reduction than the random coordinate sampling with the probabilities in (24). We thus have the following result.

Corollary 7. In expectation, SGSL-CD converges equally fast or faster than both SGS-CD and SL-CD.

We remark that we could have compared the convergence rates of SGSL-CD and SL-CD following a procedure similar to the one used to compare SGS-CD and SU-CD, where we would define a norm using non-overlapping sets (in this case, accounting also for the coordinate-wise Lipschitz consants) as an intermediate step to compare the strong convexity constants $\sigma_{\mathcal{L}}$ and σ_{SML}. We did this derivation and observed that just as it happened with σ_{SM} in eq. (17), the lower bound on σ_{SML} is not tight and suggests that SL-CD could be faster than SGSL-CD, which as we argued above, is not true.

Corollary 5 states that SL-CD is faster than SU-CD, and Corollary 7 states that SGSL-CD is the fastest of all algorithms analyzed here. However, these two methods depend on the knowledge of the coordinate-wise Lipschitz constants L_ℓ (see eq. (23)). These constants are the global upper bounds on the diagonal entries of the Hessian $H = \nabla^2 F$, given by

$$H_{\ell\ell}(\lambda) = \nabla^2 f_i^*(U_i^T \Lambda \lambda) + \nabla^2 f_j^*(U_j^T \Lambda \lambda), \ \ell \equiv (i,j),$$

i.e. $H_{\ell\ell}(\lambda) \leq L_\ell \ \forall \lambda$. We next describe a decentralized algorithm to estimate these values when they are not known. In Section VII we show that the versions of SL-CD and SGSL-CD that use estimated constants, which we call SeL-CD and SGSeL-CD, still perform remarkably well.

C. Smoothness constants estimation

In [20] the author proposed a method to estimate the value of the instantaneous Lipschitz constants $L_\ell(\lambda)$ when they are not known. By *instantaneous* we mean the value of the Lipschitz constants at the current point λ^k, and not global values valid for any value of λ.

The procedure consists on finding, every time that variable λ_ℓ is going to be updated at iteration k, the *lowest* value $L_\ell(\lambda^k)$ such that after applying update (4) with stepsize $\eta = 1/L_\ell(\lambda^k)$ it holds that $\nabla_\ell F(\lambda^k) \cdot \nabla_\ell F(\lambda^{k+1}) > 0$. In other words, the procedure searches for a Lipschitz constant (or equivalently, a stepsize) for which the update (4) does not *overshoot*, making the gradient take a completely different direction.

Algorithm 3 Online smoothness constant estimation

1: **Assumption:** Nodes i and j will update $\lambda_\ell, \ell \equiv (i,j)$ and they have already exchanged their $\nabla f_x^*(u_x^T A\lambda), x = i, j$.
Inputs: Instantaneous smoothness starting value $\widehat{L}_\ell^0$
Each node $x = i, j$ **then runs:**
2: Compute $\nabla_\ell F(\lambda^k)$ with (6) using $\nabla f_x^*(u_x^T A\lambda), x = i, j$
3: Set $\widehat{L}_\ell \leftarrow \widehat{L}_\ell^0$
4: **do ...**
5: Set $\widehat{L}_\ell \leftarrow 2 \cdot \widehat{L}_\ell$
6: Compute $\widehat{\lambda}_\ell = \lambda_\ell^k - (1/\widehat{L}_\ell) \cdot \nabla_\ell F(\lambda^k)$
7: Compute $\nabla f_x^*(u_x^T A\widehat{\lambda}_\ell)$ and send to neighbor
8: Compute $\nabla_\ell F(\widehat{\lambda})$ with (6) using $\nabla f_x^*(u_x^T A\widehat{\lambda}_\ell), x{=}i, j$
9: **... while** $\nabla_\ell F(\lambda^k) \cdot \nabla_\ell F(\widehat{\lambda}) \leq 0$
10: **end do-while**
11: Set $L_\ell^{k+1} \leftarrow 0.5 \cdot \widehat{L}_\ell$ and $\lambda_\ell^{k+1} \leftarrow \widehat{\lambda}_\ell$

The procedure to estimate $L_\ell(\lambda^k)$ is shown in Algorithm 3. In our numerical simulations, we denote SeL-CD and SGSeL-CD the versions of SL-CD and SGSL-CD that use estimated Lipschitz constants instead of the exact L values. SeL-CD is obtained by replacing line 8 in Alg. 1 with Alg. 3 and using the estimated values L_ℓ for the random sampling. SGSeL-CD is obtained by replacing line 10 in Alg. 2 with Alg. 3 and using the estimated values L_ℓ for the GSL neighbor choice.

The choice of the initial value $\widehat{L}_\ell^0$ before entering the search loop is subject to a trade-off: if $\widehat{L}_\ell^0$ is too big, the loop will be exited after only one iteration but we risk being too conservative and making a much smaller step than we could. Conversely, if $\widehat{L}_\ell^0$ is too small, by repeated doubling we will eventually find the value $\widehat{L}_\ell$ that is closest to the true instantaneous smoothness $L_\ell(\lambda^k)$, but this may take many iterations inside the loop, which means many rounds of computation and communication for the nodes involved.

How are these estimated values expected to perform with respect to the analytical ones? This depends heavily on the problem at hand. We can easily construct a case where the exact constants perform better than the estimated: assume that we are in the parallel distributed setting, where we perform primal optimization, and the function to optimize is $F(x) = x^T \text{diag}(L_1, \ldots, L_d)x$. If $x^0 \neq \mathbf{0}$, then the algorithm using the analytic constants can converge in d steps (one in each coordinate). This is actually what either SGS-CD or SGSL-CD would achieve. However, using estimated constants will most likely exit the search loop finding values $\widehat{L}_\ell \neq L_\ell$, and thus will need more iterations. Conversely, the analytical constants L_ℓ are *global* quantities, and therefore, although they are valid in the complete optimization space, they might be very different to the real instantaneous Lipschitz constants for many values of λ (or x in the example above). In that case, we may get a much better approximation to the instantaneous value using the estimations, and therefore a faster convergence due to using a larger stepsize. In Section VII-B we provide numerical tests where we observe both behaviors.

VI. Additional considerations

A. Application to parallel distributed optimization

In the parallel distributed setup, the parameter vector is stored in a server accessible by multiple workers, each of which modifies some or all of the coordinates of the parameter. We assume that coordinates are updated by a single worker at each iteration and workers always access the most recent value of the parameter.

In this setting, if there are E coordinates, n workers, and we let each worker i update a different set $\mathcal{S}_i$ of coordinates such that (i) the sets overlap, and (ii) each coordinate can be updated by exactly two workers, then all results presented previously (Theorems 1, 2, 4, and 6) hold also for this setting. We remark these two conditions are *not* necessary conditions to apply the set-wise CD algorithms to the parallel distributed setting, but only to directly apply the results of the theorems, which were derived for the decentralized setting. In fact, the family of set-wise CD algorithms *can always be applied* to the parallel distributed setting *independently* of the degree of overlapping of the sets and the number of the coordinates modified by each worker.

We can then also easily construct a setting where SGS-CD is $N_{\max}$ times faster than SU-CD: let all sets have the same size $|\mathcal{S}_i| = N_{\max}\ \forall i$, exactly $(N_{\max}-1)$ coordinates in each set have $\nabla_m F(\lambda) = 0$, and only one ℓ have $\nabla_\ell F(\lambda) \neq 0$. In this case, on average *only* $\frac{1}{N_{\max}}$ *times will SU-CD choose the coordinate that gives some improvement*, while SGS-CD will do it at all iterations.

Note that achieving the maximum speedup for this carefully crafted scenario requires that the gradients of all coordinates are independent, which is not verified in the decentralized optimization setting: according to eq. (6), for a $\nabla_m F$ to be zero, it must hold that $\nabla f_i^* = \nabla f_j^*$ for $m \equiv (i,j)$. But unless this equality holds for *all* $(i,j) \in \mathcal{E}$ (i.e., unless the minimum has been attained), λ will continue to change, and the ∇f_i^* will differ. This prevents us from easily designing a scenario in the decentralized setting where SGS-CD attains the speedup upper bound with respect to SU-CD. Nevertheless, in Figure 2 we show examples where (i) *the speedup increases linearly with* $N_{\max}$ for the decentralized setting, and (ii) *the speedup matches* $N_{\max}$ for the parallel distributed setting.

B. Dual-unfriendly functions and relation to Dual Ascent

The exposition that we have adopted up to this point may suggest that in order to run the set-wise CD methods presented here, one should be able to compute the Fenchel conjugates f_i^* for $i \in [n]$. Computing these functions may be tedious, and in some cases, like the logistic regression example presented in the next section, simply impossible.

However, we remark that the dual coordinate algorithm presented here is equivalent to the dual decomposition method (Section 2.2 in [31]) and therefore the gradients ∇f_i^* can be directly computed by minimizing the per-node Lagrangian (see also Proposition 11.3 in [32])

$$\nabla f_i^*(u_i^T A\lambda) = \arg\min_{\theta_i} \left[f_i(\theta_i) + \sum_{\ell \in \mathcal{S}_i} A_{i\ell}\lambda_\ell \theta_i \right]. \qquad (37)$$

Therefore, to apply the algorithms presented here we do *not* need to be able to compute the Fenchel conjugates f_i^*, as long as we can solve (37) analytically or numerically to a high precision. This is what we do in our experiments of Section VII for the logistic regression problem.

C. Case d > 1

To extend the proofs above for $d > 1$, the block arrays Λ and U_i should be used instead of A and u_i, and the selector matrices T_i should be redefined in the same way (i.e., by making a Kronecker product with the identity). Then, all the operations that in the proofs above are applied *per entry* (scalar coordinate) of the vector λ, should now be applied to *the magnitude* of each vector coordinate $\lambda_\ell \in \mathbb{R}^d$ of $\lambda \in \mathbb{R}^{Ed}$. Also, since $\nabla_m F \in \mathbb{R}^d$, in this case the GS rule becomes $\text{argmax}_{m\in\mathcal{S}_i} \|\nabla_m F(\lambda)\|_2^2$ (and the GSL rule is modified analogously).

VII. Numerical Results

In this section, we test the algorithms proposed in numerical simulations and analyze their performance in a range of different scenarios. In all cases, we used (37) to compute the ∇f_i^* needed in (6). For quadratic and linear least squares problems (37) has a closed-form expression, while for logistic regression we used the SciPy module for the optimization [33].

A. SU-CD vs SGS-CD: speedup increase with $N_{\max}$

Figure 2 shows an example in the decentralized setting where the speedup of SGS-CD compared to SU-CD increases linearly with $N_{\max}$ (left plots), and an example in the parallel distributed setting where SGS-CD achieves the maximum speedup of $N_{\max}$ (right plots).

For the decentralized setting, we created two regular graphs of $n = 32$ nodes and degrees $N_{\max} = 8$ and 12, respectively. The local functions were $f_i(\theta) = \theta^T c I_d \theta$ with $d = 5$, and the constant c being much larger for one node than all others. This choice gave a few edges with smoothness constants much smaller than the rest, maximizing the chances to observe the advantages of SGS-CD versus SU-CD (see also the discussion in Section 4.1 of [21]).

For the parallel distributed setting, we created a problem that was separable per-coordinate, and we tried to recreate the conditions described in Section VI-A to approximate the $N_{\max}$ gain. We chose $F(x) = x^T \text{diag}(a_1, ..., a_d)x$ with $d = 48$ and $a_i \sim \mathcal{N}(10, 3)\ \forall i$. We then created n sets of $N_{\max}$ coordinates such that each coordinate belonged to exactly two sets, and simulated two different distributions of the $d = 48$ coordinates: one with $n = 24$ sets of $N_{\max} = 4$ coordinates, and another with $n = 12$ sets of $N_{\max} = 8$ coordinates. Following the reasoning in Sec. VI-A, we set the initial value of $(N_{\max} - 1)$ coordinates in each set to $x_m^0 = 1$ (close to the optimal value $x_m^* = 0$), and the one remaining to $x_\ell^0 = 100$ (far away from $x_\ell^* = 0$).

In all plots of Fig. 2 we used the portion of the curves highlighted with thicker lines to estimate the suboptimality reduction factor $(1-\rho)$, and called ρ_U and ρ_G the rates

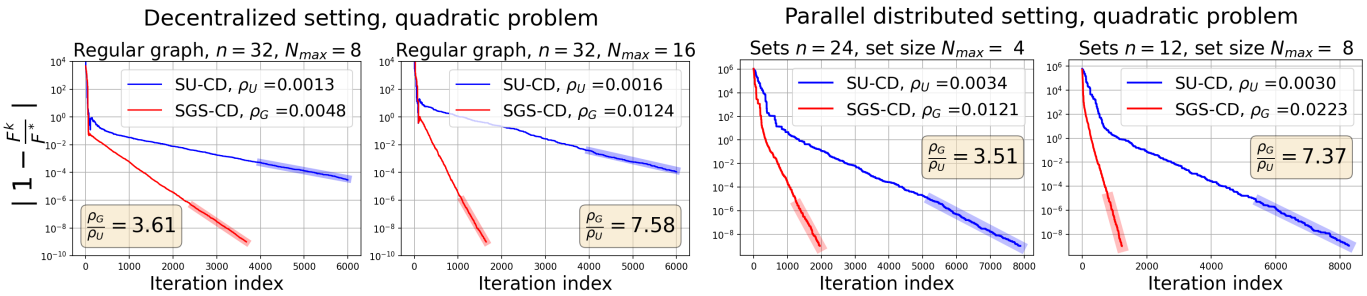


Fig. 2: Comparison of the convergence rates of SU-CD and SGS-CD for quadratic problems in two settings: decentralized optimization over a network (left plots), and parallel distributed computation with parameter server (right plots).

of SU-CD and SGS-CD, respectively. In all cases we see that $1 \leq \frac{\rho_G}{\rho_U} \leq N_{\max}$, as predicted by Theorem 2. We additionally observe that this ratio increases approximately in the same proportion as $N_{\max}$ for the decentralized setting, and is approximately equal to $N_{\max}$ in the parallel distributed.

B. Estimated vs exact Lipschitz constants

As mentioned in Section V-C, whether the estimated instantaneous Lipschitz constants $\widehat{L}_i$ achieve a faster or slower per-iteration convergence of the exact global L_i than the Lipschitz-informed algorithms depends heavily on the problem at hand. In this section, we compare SL-CD against SeL-CD in two different polynomial functions and show that the fastest algorithm is different in each case (Figure 3).

We consider the parallel distributed setting of the last plot in Fig. 2, where the parameter vector x has 48 coordinates and 12 workers update 8 coordinates each, such that each coordinate is updated by exactly two workers. We compare the performance of SL-CD and SeL-CD in two functions: a quadratic $F(x) = a_1x_1^2 + \ldots + a_dx_d^2 + 1$ and an order-four polynomial $F(x) = a_1x_1^4 + \ldots + a_dx_d^4 + 1$. For the former, the global (and also the instantaneous) coordinate-wise Lipschitz constants are $\{L_i\} = \{2a_i\}$, while for the latter they are $\{L_i\} = \{12a_i(\hat{x}_i)^2)\}$, where $\hat{x}_i$ is the maximum absolute value taken by the entry x_i throughout the optimization. The constants a_i were set to random integers sampled at uniform in the interval [1,100].

Fig. 3 shows the performance of SL-CD and SeL-CD in the quadratic (left plot) and the order-four (right plot) problems. As expected, SL-CD converges faster than SeL-CD in the quadratic problem, where once a coordinate is selected for the first time it is set to its optimal value in that single iteration. Note also that due to this behavior and the fact that the coordinate sampling probabilities of SL-CD are fixed (cf. eq. (24)), sampling a not-yet-optimized coordinate becomes more and more difficult as the iterations progress, which is why the convergence of SL-CD in the left plot of Fig. 3 shows a stairwise pattern where the length of the steps becomes larger with the iterations. The estimated constants $\widehat{L}_\ell$ approximate the optimal values $2a_i$ as well as possible, but cannot match them exactly and thus SeL-CD converges more slowly.

Conversely, in the order-four function SeL-CD converges faster than SL-CD, since in this case the estimated constants will always be closer to the true instantaneous coordinate-wise Lipschitz values than the global smoothness constants

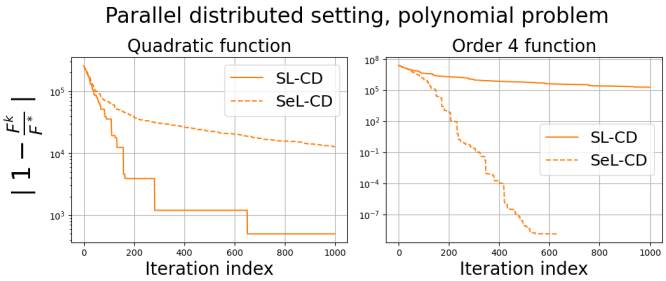


Fig. 3: Comparison of the performance of SL-CD (using global exact L_i values) and SeL-CD in the parallel distributed setting. The experiment was designed to show that which of the two methods is faster depends on the problem considered.

L_ℓ. Note, however, that if the exact instantaneous values could be known at each iteration (which is the case of the quadratic function in the left plot of Fig. 3), SL-CD using these values would always be faster than SeL-CD.

C. Number of iterations vs communication complexity

Figure 4 shows the performance of all algorithms proposed for the linear least squares problem

$$f_i(\theta) = \frac{1}{M}\|X_i\theta - Y_i\|_2^2, \; X_i \in \mathbb{R}^{M\times d}, \; Y_i \in \mathbb{R}^M,$$

in two random graphs of $n = 32$ nodes and link probabilities of 0.1 (left plots) and 0.5 (right plots), respectively. The data was generated with the model of [11], $d = 5, M = 30$, and the Y values were additionally multiplied by the index of the corresponding node to have non-iid data between the nodes. Here we do not only show the convergence of the algorithms in terms of the number of iterations (top plots) but also in terms of the number of vectors in $\mathbb{R}^d$ transmitted through the network for each suboptimality value computed. Table II shows the communication complexity of each algorithm in these terms.

TABLE II: Communication complexity of each algorithm: number of vectors in $\mathbb{R}^d$ transmitted in one iteration for an arbitrary activated node i. Variable I indicates the number of iterations inside the do-while loop in Alg. 3.

SU-CD	2	SGS-CD	$N_i + 1$
SL-CD	2	SGSL-CD	$N_i + 1$
SeL-CD	$2 + 2I$	SGSeL-CD	$N_i + 1 + 2I$

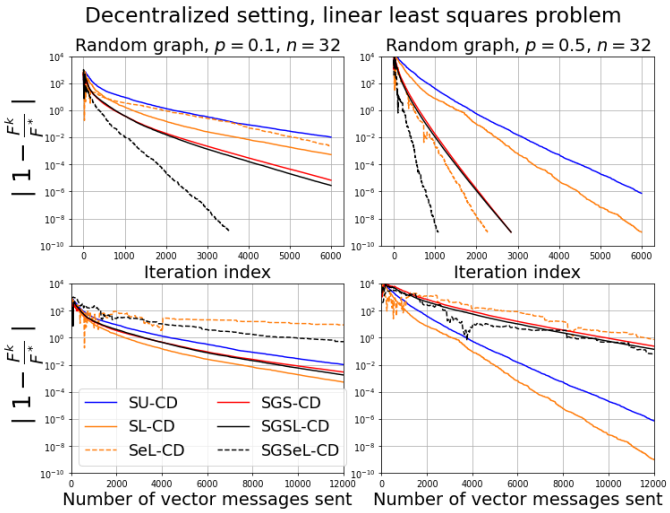


Fig. 4: Performance of the algorithms presented in a linear least squares problem and two random graphs with different numbers of edges (left and right columns). The top plots show convergence in terms of the number of iterations, and the bottom plots, in terms of the number of vectors in $\mathbb{R}^d$ transmitted.

In terms of the number of iterations, we confirm the conclusions of all our corollaries, namely (i) SGS-CD converges faster than SU-CD (Corollary 3), (ii) SL-CD converges faster than SU-CD (Corollary 5), and (iii) SGSL-CD converges faster than both SL-CD and SGS-CD (Corollary 7). Whether the versions with estimated Lipschitz constants SeL-CD and SGSeL-CD are faster than their counterparts with exact Lipschitz knowledge SL-CD and SGSL-CD depends on the problem instance, as discussed in the previous section. As already observed in Fig. 2, the speedup of the algorithms applying either the GS or the GSL rule increases radically as the graph becomes more connected.

The plots in terms of the number of vectors transmitted provide a complementary point of view of the relative performance of all these algorithms. We observe that for a sparsely connected graph (bottom-left plot in Fig. 4), SGS-CD and SGSL-CD may still achieve lower suboptimality than SU-CD for the same number of transmissions, but they are already outperformed by SL-CD. Algorithms SeL-CD and SGSeL-CD are the slowest when plotted against number of transmissions, since they have the additional overhead of estimating the smoothness constants. The gap between SU-CD and SL-CD, which are the algorithms with the lowest number of vector transmissions per iteration (see Table II) and the rest of the algorithms becomes larger (in favor of the former) as the graph becomes more connected (bottom-left plot in Fig. 4). While it is natural that plotting the suboptimality reduction versus the number of vector transmissions benefits the algorithms using randomized neighbor selection (and no smoothness estimation), this does not necessarily mean that they will converge faster in a real system in terms of wall-clock time. We discuss this in more detail in Section VIII.

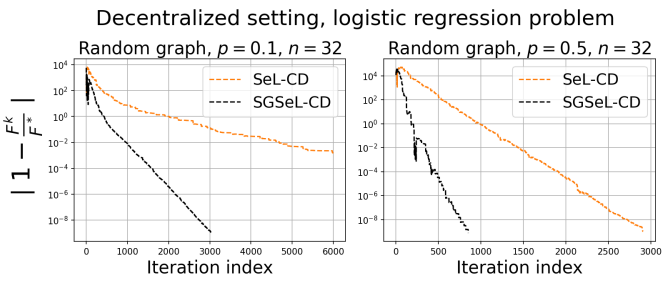


Fig. 5: Convergence of SeL-CD and SGSeL-CD in a logistic regression problem.

D. A dual-unfriendly problem with no L *knowledge*

Figure 5 shows the convergence of SeL-CD and SGSeL-CD in the logistic regression problem

$$f_i(\theta) = \frac{1}{M}\sum_{j=1}^{M} \log\left(1 + \exp(-[Y_i]_j \cdot ([X_i]_j)^T \theta_i)\right) + c\,\|\theta_i\|_2^2$$

where $[Y_i]_j$ and $([X_i]_j)^T$ are the j-th component and the j-th row of arrays $Y_i \in \mathbb{R}^M$ and $X_i \in \mathbb{R}^{M\times d}$, respectively. We ran the simulation for the same graphs and parameter choices used for the experiments in Section VII-C. In this case, we cannot compute analytically the optimal value of (37), so we did the optimization in (37) using the SciPy module. For the same reason, we do not know the true coordinate Lipschitz values L_ℓ, so we test only the algorithms using estimated constants.

As in the previous examples, both algorithms converge linearly, and SGSeL-CD is faster than SeL-CD. We may remark, however, that the gap between the two algorithms does not increase with the graph connectivity, as observed between SL-CD and SGSL-CD in the top plots of Fig. 4. Indeed, we observed that when the L_ℓ are estimated, the gap between SGSeL-CD and SeL-CD may or may not increase with the connectivity of the graph. We attribute this effect to the fact that the performance of the algorithms depends very much on how close to optimal the estimated Lipschitz constants are, and therefore, some instances that allow for a better fit of the true constants using Alg. 3 have advantage over others whose true constants cannot be well approximated. In the next section we discuss how to improve the estimation of Alg. 3, and the associated costs of this improvement.

VIII. Discussion and Conclusion

We have presented the class of *set-wise CD* optimization algorithms, where in a multi-agent system workers are allowed to modify only a subset of the total number of coordinates at each iteration. These algorithms are suitable for (dual) asynchronous decentralized optimization and (primal) distributed parallel optimization.

We studied the convergence of a number of set-wise CD variants: random uniform and Gauss-Southwell set-wise coordinate selection (SU-CD and SGS-CD), and their Lipschitz-informed versions (SL-CD and SGSL-CD). We showed linear convergence for all variants for smooth and strongly convex functions f_i, which required developing a new methodology that extends previous results on CD methods.

In particular, we proved that in expectation, when convergence is measured *in terms of number of iterations*, both SGS-CD and SL-CD are faster than SU-CD, and SGSL-CD is the fastest of them all. However, running one iteration of each algorithm requires different amount of computation and communication (see Table II). When the convergence is measured *in terms of number of vectors transmitted* through the network, the random algorithms become better than the ones based on the GS rule as the connectivity of the network increases. However, neither the performance measured against number of iterations nor of vectors transmitted is sufficient to decide which will perform the fastest in a real setup. Since all algorithms are asynchronous and can modify multiple coordinates simultaneously, in a real scenario many iterations and vector transmissions will occur at the same time, and the actual wall-clock time of the algorithm will depend on the network connectivity and where the bottlenecks of the system are (e.g. low link capacities, presence of stragglers).

Lastly, we proposed the methods SeL-CD and SGSeL-CD, which run respectively SL-CD and SGSL-CD but with online coordinate Lipschitz constant estimation for the cases where these values cannot be easily computed. While these estimations can achieve remarkably good performance in terms of number of iterations (top plots in Fig. 4), they come with a greater penalty in the number of vectors transmitted, which shows clearly in the bottom plots of Fig. 4. In cases where there is no alternative but to use estimated Lipschitz constants, as in our logistic regression example of Section VII-D, and if the communicaion constraints allow it, we could design an algorithm better than Alg. 3 that, instead of doubling $\widehat{L}_\ell$ at each time (line 5), runs a bisection search to approximate the true instantaneous L_ℓ as much as possible. This would, of course, increase the communication costs even more; whether this penalty is worth paying is an engineering decision that will depend on the system setup and its constraints.

REFERENCES

[1] X. Lian, C. Zhang, H. Zhang, C.-J. Hsieh, W. Zhang, and J. Liu, "Can decentralized algorithms outperform centralized algorithms? a case study for decentralized parallel stochastic gradient descent," *Advances in Neural Information Processing Systems*, vol. 30, 2017.

[2] P. Wan and M. D. Lemmon, "Event-triggered distributed optimization in sensor networks," in *2009 International Conference on Information Processing in Sensor Networks*, pp. 49–60, IEEE, 2009.

[3] M. Alrowaily and Z. Lu, "Secure edge computing in IoT systems: review and case studies," in *2018 IEEE/ACM Symposium on Edge Computing (SEC)*, pp. 440–444, IEEE, 2018.

[4] S. Warnat-Herresthal, H. Schultze, K. L. Shastry, S. Manamohan, S. Mukherjee, V. Garg, R. Sarveswara, K. Händler, P. Pickkers, N. A. Aziz, *et al.*, "Swarm learning for decentralized and confidential clinical machine learning," *Nature*, vol. 594, no. 7862, pp. 265–270, 2021.

[5] A. Nedic and A. Ozdaglar, "On the rate of convergence of distributed subgradient methods for multi-agent optimization," in *2007 46th IEEE Conference on Decision and Control*, pp. 4711–4716, IEEE, 2007.

[6] G. Neglia, C. Xu, D. Towsley, and G. Calbi, "Decentralized gradient methods: does topology matter?," in *International Conference on Artificial Intelligence and Statistics*, pp. 2348–2358, PMLR, 2020.

[7] D. Jakovetic, D. Bajovic, A. K. Sahu, and S. Kar, "Convergence rates for distributed stochastic optimization over random networks," in *2018 IEEE Conference on Decision and Control (CDC)*, pp. 4238–4245, IEEE, 2018.

[8] X. Lian, W. Zhang, C. Zhang, and J. Liu, "Asynchronous decentralized parallel stochastic gradient descent," in *International Conference on Machine Learning*, pp. 3043–3052, PMLR, 2018.

[9] K. Yuan, Q. Ling, and W. Yin, "On the convergence of decentralized gradient descent," *SIAM Journal on Optimization*, vol. 26, no. 3, pp. 1835–1854, 2016.

[10] W. Shi, Q. Ling, G. Wu, and W. Yin, "EXTRA: An exact first-order algorithm for decentralized consensus optimization," *SIAM Journal on Optimization*, vol. 25, no. 2, pp. 944–966, 2015.

[11] K. Scaman, F. Bach, S. Bubeck, Y. T. Lee, and L. Massoulié, "Optimal algorithms for smooth and strongly convex distributed optimization in networks," in *international conference on machine learning*, pp. 3027–3036, PMLR, 2017.

[12] H. Hendrikx, F. Bach, and L. Massoulié, "Accelerated decentralized optimization with local updates for smooth and strongly convex objectives," in *The 22nd International Conference on Artificial Intelligence and Statistics*, pp. 897–906, PMLR, 2019.

[13] C. A. Uribe, S. Lee, A. Gasnikov, and A. Nedić, "A dual approach for optimal algorithms in distributed optimization over networks," in *2020 Information Theory and Applications Workshop (ITA)*, pp. 1–37, IEEE, 2020.

[14] F. Iutzeler, P. Bianchi, P. Ciblat, and W. Hachem, "Asynchronous distributed optimization using a randomized alternating direction method of multipliers," in *52nd IEEE Conference on Decision and Control (CDC)*, pp. 3671–3676, IEEE, 2013.

[15] E. Wei and A. Ozdaglar, "On the $O(1/k)$ convergence of asynchronous distributed alternating direction method of multipliers," in *2013 IEEE Global Conference on Signal and Information Processing*, pp. 551–554, IEEE, 2013.

[16] J. Xu, S. Zhu, Y. C. Soh, and L. Xie, "Convergence of asynchronous distributed gradient methods over stochastic networks," *IEEE Transactions on Automatic Control*, vol. 63, no. 2, pp. 434–448, 2017.

[17] K. Srivastava and A. Nedic, "Distributed asynchronous constrained stochastic optimization," *IEEE Journal of Selected Topics in Signal Processing*, vol. 5, no. 4, pp. 772–790, 2011.

[18] S. S. Ram, A. Nedić, and V. V. Veeravalli, "Asynchronous gossip algorithms for stochastic optimization," in *Proceedings of the 48h IEEE Conference on Decision and Control (CDC) held jointly with 2009 28th Chinese Control Conference*, pp. 3581–3586, IEEE, 2009.

[19] M. Costantini, N. Liakopoulos, P. Mertikopoulos, and T. Spyropoulos, "Pick your neighbor: Local gauss-southwell rule for fast asynchronous decentralized optimization," in *2022 IEEE 61st Conference on Decision and Control (CDC)*, pp. 1602–1609, IEEE, 2022.

[20] Y. Nesterov, "Efficiency of coordinate descent methods on huge-scale optimization problems," *SIAM Journal on Optimization*, vol. 22, no. 2, pp. 341–362, 2012.

[21] J. Nutini, M. Schmidt, I. Laradji, M. Friedlander, and H. Koepke, "Coordinate descent converges faster with the Gauss-Southwell rule than random selection," in *International Conference on Machine Learning*, pp. 1632–1641, PMLR, 2015.

[22] J. Nutini, I. Laradji, and M. Schmidt, "Let's make block coordinate descent go fast: Faster greedy rules, message-passing, active-set complexity, and superlinear convergence," *arXiv preprint arXiv:1712.08859*, 2017.

[23] J. Tsitsiklis, D. Bertsekas, and M. Athans, "Distributed asynchronous deterministic and stochastic gradient optimization algorithms," *IEEE Transactions on Automatic Control*, vol. 31, no. 9, pp. 803–812, 1986.

[24] Z. Peng, Y. Xu, M. Yan, and W. Yin, "Arock: an algorithmic framework for asynchronous parallel coordinate updates," *SIAM Journal on Scientific Computing*, vol. 38, no. 5, pp. A2851–A2879, 2016.

[25] L. Xiao, A. W. Yu, Q. Lin, and W. Chen, "DSCOVR: Randomized primal-dual block coordinate algorithms for asynchronous distributed optimization," *The Journal of Machine Learning Research*, vol. 20, no. 1, pp. 1634–1691, 2019.

[26] S. Pu, W. Shi, J. Xu, and A. Nedić, "Push–pull gradient methods for distributed optimization in networks," *IEEE Transactions on Automatic Control*, vol. 66, no. 1, pp. 1–16, 2020.

[27] A. Verma, M. M. Vasconcelos, U. Mitra, and B. Touri, "Maximal dissent: a state-dependent way to agree in distributed convex optimization," *IEEE Transactions on Control of Network Systems*, 2023.

[28] Y. You, X. Lian, J. Liu, H.-F. Yu, I. S. Dhillon, J. Demmel, and C.-J. Hsieh, "Asynchronous parallel greedy coordinate descent," *Advances in Neural Information Processing Systems*, vol. 29, 2016.

[29] S. Boyd and L. Vandenberghe, *Convex optimization*. Cambridge university press, 2004.

[30] S. J. Wright, "Coordinate descent algorithms," *Mathematical Programming*, vol. 151, no. 1, pp. 3–34, 2015.

[31] S. Boyd, N. Parikh, E. Chu, B. Peleato, J. Eckstein, *et al.*, "Distributed optimization and statistical learning via the alternating direction method

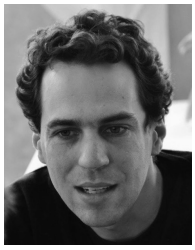
of multipliers," *Foundations and Trends® in Machine learning*, vol. 3, no. 1, pp. 1–122, 2011.

[32] R. T. Rockafellar and R. J.-B. Wets, *Variational analysis*, vol. 317. Springer Science & Business Media, 2009.

[33] P. Virtanen, R. Gommers, T. E. Oliphant, M. Haberland, T. Reddy, D. Cournapeau, E. Burovski, P. Peterson, W. Weckesser, J. Bright, S. J. van der Walt, M. Brett, J. Wilson, K. J. Millman, N. Mayorov, A. R. J. Nelson, E. Jones, R. Kern, E. Larson, C. J. Carey, İ. Polat, Y. Feng, E. W. Moore, J. VanderPlas, D. Laxalde, J. Perktold, R. Cimrman, I. Henriksen, E. A. Quintero, C. R. Harris, A. M. Archibald, A. H. Ribeiro, F. Pedregosa, P. van Mulbregt, and SciPy 1.0 Contributors, "SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python," *Nature Methods*, vol. 17, pp. 261–272, 2020.

Marina Costantini received her Electronic Engineering degree (with honors) from the National University of La Plata, Argentina, in 2015 and her M.Sc. in communication systems from EURECOM, Sophia Antipolis, France, in 2019. She is currently pursuing a Ph.D. degree in engineering at EURECOM and Sorbonne University. Her research interests span different aspects of mathematical optimization theory, with particular interest in multi-agent optimization for distributed computing environments.

Nikolaos Liakopoulos received the Diploma degree in physics and the M.Sc. degree in control and computing from the National and Kapodistrian University of Athens, and the Ph.D. degree in communication systems from Sorbonne University in 2019. He was with the Mathematical and Algorithmic Sciences Lab, Paris Research Center, Huawei Technologies SASU, from 2016 until 2021. He is currently with Amazon Luxembourg, where he is Senior Research Scientist in Amazon Transportation Services.



Panayotis Mertikopoulos is a principal researcher at the French National Center for Scientific Research (CNRS). After completing his undergraduate studies in physics at the University of Athens, he received his MSc and MPhil degrees in mathematics from Brown University in 2005, and his PhD degree from the University of Athens in 2010. Before joining the CNRS in 2011, he spent one year as a post-doctoral fellow at École Polytechnique in Paris. Since 2011, he has held visiting or part-time positions at UC Berkeley and EPFL. His research interests span the interface of game theory, learning and optimization, with a special view towards their applications to machine learning, data science, and operations research. His most recent distinctions include a nomination for the médaille de bronze of the CNRS in computer science in 2020 and the INFORMS best paper award in network science in 2022.

Thrasyvoulos Spyropoulos received the Diploma in Electrical and Computer Engineering from the National Technical University of Athens, Greece, and a Ph.D degree in Electrical Engineering from the University of Southern California. He was a post-doctoral researcher at INRIA and then, a senior researcher with the Swiss Federal Institute of Technology (ETH) Zurich. He has served as an Associate Professor (2010-2020) and Professor (2020-2022) in EURECOM, Sophia-Antipolis, France. He is currently a Full Professor at the Technical University of Crete, Greece. He is the co-recipient of the best paper award in IEEE SECON 2008, and IEEE WoWMoM 2012, and best paper award runner-up for ACM Mobihoc 2011, IEEE WoWMoM 2015, 2021.