Accelerated Decentralized Optimization with Local Updates for Smooth and Strongly Convex Objectives

Hadrien Hendrikx
INRIA
École Normale Supérieure
MSR-INRIA Joint Centre

Francis Bach
INRIA
École Normale Supérieure

Laurent Massoulié
INRIA
École Normale Supérieure
MSR-INRIA Joint Centre

Abstract

In this paper, we study the problem of minimizing a sum of smooth and strongly convex functions split over the nodes of a network in a decentralized fashion. We propose the algorithm ESDACD, a decentralized accelerated algorithm that only requires local synchrony. Its rate depends on the condition number $\kappa$ of the local functions as well as the network topology and delays. Under mild assumptions on the topology of the graph, ESDACD takes a time $O((\tau_{\text{max}} + \Delta_{\text{max}})\sqrt{\kappa}/\gamma \ln(\epsilon^{-1}))$ to reach a precision $\epsilon$ where $\gamma$ is the spectral gap of the graph, $\tau_{\text{max}}$ the maximum communication delay and $\Delta_{\text{max}}$ the maximum computation time. Therefore, it matches the rate of SSDA (Scaman et al., 2017), which is optimal when $\tau_{\text{max}} = \Omega(\Delta_{\text{max}})$. Applying ESDACD to quadratic local functions leads to an accelerated randomized gossip algorithm of rate $O(\sqrt{\theta_{\text{gossip}}}/n)$ where $\theta_{\text{gossip}}$ is the rate of the standard randomized gossip (Boyd et al., 2006). To the best of our knowledge, it is the first asynchronous gossip algorithm with a provably improved rate of convergence of the second moment of the error. We illustrate these results with experiments in idealized settings.

1 Introduction

Many modern machine learning applications require to process more data than one computer can handle, thus forcing to distribute work among computers linked by a network. In the typical machine learning setup, the function to optimize can be represented as a sum of local functions $f(x) = \sum_{i=1}^{n} f_i(x)$, where each $f_i$ represents the objective over the data stored at node $i$. This problem is usually solved incrementally by alternating rounds of gradient computations and rounds of communications (Nedic and Ozdaglar, 2009; Boyd et al., 2011; Duchi et al., 2012; Shi et al., 2015; Mokhtari and Ribeiro, 2016; Scaman et al., 2017; Nedic et al., 2017).

Most approaches assume a centralized network with a master-slave architecture in which workers compute gradients and send it back to a master node that aggregates them. There are two main different flavors of algorithms in this case, whether the algorithm is based on stochastic gradient descent (Zinkevich et al., 2010; Recht et al., 2011) or randomized coordinate descent (Nesterov, 2012; Liu and Wright, 2015; Liu et al., 2015; Fercoq and Richtárik, 2015; Hannah et al., 2018). Although this approach usually works best for small networks, the central node represents a bottleneck both in terms of communications and computations. Besides, such architectures are not very robust since the failure of the master node makes the whole system fail. In this work, we focus on decentralized architectures in which nodes only perform local computations and communications. These algorithms are generally more scalable and more robust than their centralized counterparts (Lian et al., 2017a). This setting can be used to handle a wide variety of tasks (Colin et al., 2016), but it has been particularly studied for stochastic gradient descent, with the D-PSGD algorithm (Nedic and Ozdaglar, 2009; Ram et al., 2009, 2010) and its extensions (Lian et al., 2017b; Tang et al., 2018).

A popular way to make first order optimization faster is to use Nesterov acceleration (Nesterov, 2013). Accelerated gradient descent in a dual formulation yields optimal synchronous algorithms in the decentralized setting (Scaman et al., 2017; Ghadimi et al., 2013). Variants of accelerated gradient descent include the
acceleration of the coordinate descent algorithm (Nesterov, 2012; Allen-Zhu et al., 2016; Nesterov and Stich, 2017), that we use in this paper to solve the problem in Scaman et al. (2017). This approach yields different algorithms in which updates only involve two neighboring nodes instead of the full graph. Our algorithm can be interpreted as an accelerated version of Gower and Richtárik (2015); Necoara et al. (2017). Updates consist in gossiping gradients along edges that are sequentially picked from the same distribution independently from each other.

Using coordinate descent methods on the dual allows to have local gradient updates. Yet, the algorithm also needs to perform a global contraction step involving all nodes. In this paper, we introduce Edge Synchronous Dual Accelerated Coordinate Descent (ESDACD), an algorithm that takes advantage of the acceleration speedup in a decentralized setting while requiring only local synchrony. This weak form of synchrony consists in assuming that a given node can only perform one update at a time, and that for a given node, updates have to be performed in the order they are sampled. It is called the randomized or asynchronous setting in the gossip literature (Boyd et al., 2006), as opposed to the synchronous setting in which all nodes perform one update at each iteration. Following this convention, we may call ESDACD an asynchronous algorithm. The locality of the algorithm allows parameters to be fine-tuned for each edge, thus giving it a lot of flexibility to handle settings in which the nodes have very different characteristics.

Synchronous algorithms force all nodes to be updated the same number of times, which can be a real problem when some nodes, often called stragglers are much slower than the rest. Yet, we show that we match (up to a constant factor) the speed rates of optimal synchronous algorithms such as SSDA (Scaman et al., 2017) even in idealized homogeneous settings in which nodes never wait when performing synchronous algorithms. In terms of efficiency, we match the oracle complexity of SSDA with lower communication cost. This extends a result that is well-known in the case of averaging, i.e., that randomized gossip algorithms match the rate of synchronous ones (Boyd et al., 2006). We also exhibit a clear experimental speedup when the distributions of nodes computing power and local smoothnesses have a high variance.

Choosing quadratic $f_i$ functions leads to solving the distributed average consensus problem, in which each node has a variable $c_i$ and for which the goal is to find the mean of all variables $\bar{c} = \frac{1}{n} \sum_{i=1}^{n} c_i$. It is a historical problem (DeGroot, 1974; Chatterjee and Seneta, 1977) that still attracts attention (Cao et al., 2006; Boyd et al., 2006; Loizou and Richtárik, 2018) with many applications for averaging measurements in sensor networks (Xiao et al., 2005) or load balancing (Diekmann et al., 1999). Fast synchronous algorithms to solve this problem exist (Oreshkin et al., 2010) but no asynchronous algorithms match their rates. We show that ESDACD is faster at solving distributed average consensus than standard asynchronous approaches (Boyd et al., 2006; Cao et al., 2006) as well as more recent ones (Loizou and Richtárik, 2018) that do not show improved convergence rates for the second moment of the error. The complexity of gossip algorithms generally depends on the smallest non-zero eigenvalue of the gossip matrix $W$, a symmetric semi-definite positive matrix of size $n \times n$ ruling how nodes aggregate the values of their neighbors such that $\text{Ker}(W) = \text{Vec}(\mathbb{1})$ where $\mathbb{1}$ is the constant vector. We improve the rate from $\lambda_{\min}^+(W)$ to $O\left(\frac{1}{\sqrt{n}} \sqrt{\lambda_{\min}^+(W)}\right)$ where $\lambda_{\min}^+(W) \leq \frac{1}{\tau}$ is the smallest non-zero eigenvalue of the gossip matrix, thus gaining several orders of magnitude in terms of scaling for sparse graphs. In particular, in well-studied graphs such as the grid, we match (up to logarithmic factors that we do not consider) the $O(n^{3/2})$ iterations complexity of advanced gossip algorithms presented by Dimakis et al. (2010).

2 Model

The communication network is represented by a graph $\mathcal{G} = (V, E)$. When clear from the context, $E$ will also be used to designate the number of edges. Each node $i$ has a local function $f_i$ on $\mathbb{R}^d$ and a local parameter $x_i \in \mathbb{R}^d$. The global cost function is the sum of the functions at all nodes: $F(x) = \sum_{i=1}^{n} f_i(x_i)$ Each $f_i$ is assumed to be $L_i$-smooth and $\sigma_i$-strongly convex, which means that for all $x, y \in \mathbb{R}^d$:

\begin{align}
    f_i(x) - f_i(y) &\leq \nabla f_i(y)^T (x - y) + \frac{L_i}{2} \|x - y\|^2 \quad \text{(1)} \\
    f_i(x) - f_i(y) &\geq \nabla f_i(y)^T (x - y) + \frac{\sigma_i}{2} \|x - y\|^2. \quad \text{(2)}
\end{align}

Note that the fenchel conjugate $f^*_i$ of $f_i$ (defined in Equation (8)) is $(L^{-1}_i)$-strongly convex and $(\sigma^{-1}_i)$-smooth, as shown in Kakade et al. (2009). We denote $L_{\text{max}} = \max_i L_i$ and $\sigma_{\text{min}} = \min_i \sigma_i$. Then, we denote $\kappa_i = \frac{L_{\text{max}}}{\sigma_{\text{min}}}$, $\kappa_i$ is an upper bound of the condition number of all $f_i$, as well as an upper bound of the global condition number. Adding the constraint that all nodes should eventually agree on the final solution, so the optimization problem can be cast as:

$$\min_{x \in \mathbb{R}^n \times d; \ x_i = x, \ \forall i \in \{1, \ldots, n\}} F(x). \quad \text{(3)}$$

We assume that a communication between nodes $i, j \in V$ takes a time $\tau_{ij}$. If $(i, j) \notin E$, the communication is impossible so $\tau_{ij} = \infty$. Node $i$ takes time $\Delta_i$ to compute its local gradient.
3 Algorithm

In this section, we specify the Edge Synchronous Decentralized Accelerated Coordinate Descent (ESDACD) algorithm. We first give a formal version in Algorithm 1 and prove its convergence rate. Then, we present the modifications obtained to implement the implementable version given by Algorithm 2.

3.1 Problem derivation

In order to obtain the algorithm, we consider a matrix \( A \in \mathbb{R}^{n \times E} \) such that \( \text{Ker}(A^T) = \text{Vec}(I) \) where \( I = \sum_{i=1}^{n} e_i \) and \( e_i \in \mathbb{R}^{n \times 1} \) is the unit vector of size \( n \) representing node \( i \). Similarly, we will denote \( e_{ij} \in \mathbb{R}^{E \times 1} \) the unit vector of size \( E \) representing coordinate \((i, j)\). Then, the constraint in Equation (3) can be expressed as \( A^T x = 0 \) because if \( x \in \text{Ker}(A^T) \) then all its coordinates are equal and the problem writes:

\[
\min_{x \in \mathbb{R}^{n \times d}} \quad F(x). \tag{4}
\]

This problem is equivalent to the following one:

\[
\min_{x \in \mathbb{R}^{n \times d}} \max_{\lambda \in \mathbb{R}^{E \times d}} \quad F(x) - \langle \lambda, A^T x \rangle, \tag{5}
\]

where the scalar product is the usual scalar product over matrices \( \langle x, y \rangle = \text{Tr}(x^T y) \) because the value of the solution is infinite whenever the constraint is not met. This problem can be rewritten:

\[
\max_{\lambda \in \mathbb{R}^{E \times d}} \min_{x \in \mathbb{R}^{n \times d}} \quad F(x) - \langle \lambda, x \rangle \tag{6}
\]

because \( F \) is convex and \( A^T I = 0 \). Then, we obtain the dual formulation of this problem, which writes:

\[
\max_{\lambda \in \mathbb{R}^{E \times d}} \quad -F^*(A\lambda), \tag{7}
\]

where \( F^* \) is the Fenchel conjugate of \( F \) which is obtained by the following formula:

\[
F^*(y) = \max_{x \in \mathbb{R}^{n \times d}} \langle x, y \rangle - F(x). \tag{8}
\]

\( F^* \) is well-defined and finite for all \( y \in \mathbb{R}^{E \times d} \) because \( F \) is strongly convex. We solve this problem by applying a coordinate descent method. If we denote \( F_A^* : \lambda \to F^*(A\lambda) \) then the gradient of \( F_A^* \) in the direction \((i, j)\) is equal to \( \nabla_{ij} F_A^* = e_{ij}^T A^T \nabla F^* \). Therefore, the sparsity pattern of \( A \) will determine how many nodes are involved in a single update. Since we would like to have local updates that only involve the nodes at the end of a single edge, we choose \( A \) such that, for any \( \mu_{ij} \in \mathbb{R} \):

\[
Ae_{ij} = \mu_{ij}(e_i - e_j). \tag{9}
\]

This choice of \( A \) satisfies \( e_{ij}^T A^T I = 0 \) for all \((i, j) \in E \) and \( \text{Ker}(A^T) \subset \text{Vec}(I) \) as long as \((V, E_+)\) is connex where \( E_+ = \{(i, j) \in E, \mu_{ij} > 0\} \). Such \( A \) happens to be canonical since it is a square root of the Laplacian matrix if all \( \mu_{ij} \) are chosen to be equal to 1. When not explicitly stated, all \( \mu_{ij} \) are assumed to be constant so that \( A \) only reflects the graph topology. Other choices of \( A \) involving more than two nodes per row are possible and would change the trade-off between the communication cost and computation cost but they are beyond the scope of this paper.

3.2 Formal algorithm

The algorithm can then be obtained by applying ACDM (Nesterov and Stich, 2017) on the dual formulation. We need to define several quantities. Namely, we denote \( p_{ij} \in \mathbb{R} \) the probability of selecting edge \((i, j)\) and \( \sigma_A \in \mathbb{R} \) the strong convexity of \( F_A^* \). \( A^+ \in \mathbb{R}^{E \times n} \) is the pseudo-inverse of \( A \) and \( \|x\|^2_{A^+ A} = x^T A^+ A x \) for \( x \in \mathbb{R}^{E \times 1} \). Variable \( S \in \mathbb{R} \) is such that for all \((i, j) \in E \),

\[
e_{ij}^T A^+ A e_{ij} \mu_{ij}^2 p_{ij}^2 (\sigma_i^{-1} + \sigma_j^{-1}) \leq S^2.
\]

We define \( \delta = \theta \frac{1 - \theta}{1 + \theta} \in \mathbb{R} \) with

\[
\theta^2 = \min_{i, j} \quad \frac{p_{ij}^2}{\mu_{ij}^2 e_{ij}^T A^T A e_{ij}} \frac{\sigma_A}{\sigma_i^{-1} + \sigma_j^{-1}} \geq \frac{\sigma_A}{S^2}. \tag{10}
\]

Finally, \( \eta_{ij} = \frac{1}{1 + \theta} \left( \mu_{ij}^2 (\sigma_i^{-1} + \sigma_j^{-1})^{-1} + (p_{ij} S^2)^{-1} \right) \in \mathbb{R} \) and

\[
g_{ij}(y_t) = e_{ij}^T A^T \nabla F^*(Ay_t) \in \mathbb{R}^{E \times d}. \tag{11}
\]

**Algorithm 1** Asynchronous Decentralized Accelerated Coordinate Descent

\[
y_0 = 0, \quad v_0 = 0, \quad t = 0
\]

**while** \( t < T \)**

\[
\text{Sample } (i, j) \text{ with probability } p_{ij}
\]

\[
y_{t+1} = (1 - \delta) y_t + \delta v_t - \eta_{ij} g_{ij}(y_t)
\]

\[
v_{t+1} = (1 - \theta) v_t + \theta y_t - \frac{\theta}{\sigma_A} g_{ij}(y_t)
\]

**end while**

**Theorem 1**. Let \( y_t \) and \( v_t \) be the sequences generated by Algorithm 1. Then:

\[
2 \mathbb{E} [F_A^*(x_t)] - F_A^*(x^*) + \sigma_A \mathbb{E} [v_t^2] \leq C (1 - \theta)^t, \tag{12}
\]

with \( x_t = (1 + \theta) y_t - \theta v_t, \quad x^* = \arg \min_x F_A^*(x), \quad r_t^2 = \|v_t - x^*\|^2_{A^+ A} \) and \( C = r_0^2 + 2 (F_A^*(x_0) - F_A^*(x^*)) \).

Theorem 1 shows that Algorithm 1 converges with rate \( \theta \). Lemma 4, in Appendix C shows that

\[
\sigma_A \geq \frac{\lambda_{\min}^+ (A^T A)}{L_{\max}}, \tag{13}
\]

where \( \lambda_{\min}^+ (A^T A) \in \mathbb{R} \) is the smallest eigenvalue of \( A^T A \). The condition number of the problem then
appears in the \( L_{\text{max}} (\sigma_i^{-1} + \sigma_j^{-1}) \) term whereas the other terms are strictly related to the topology of the graph. Parameter \( \theta \) is invariant to the scale of \( \mu \) because rescaling \( \mu \) would also multiply \( \lambda^+ \min (A^T A) \) by the same constant. The \( p_{ij}^2 / (\sigma_i^{-1} + \sigma_j^{-1}) \) term indicates that non-smooth edges should be sampled more often, and the square root dependency is consistent with known results for accelerated coordinate descent methods (Allen-Zhu et al., 2016; Nesterov and Stich, 2017). If both sampling probabilities and smoothnesses are fixed, the \( \mu_{ij} \) terms can be used to make the dual coordinate (which corresponds to the edge) smoother so that larger step sizes can be used to compensate for the fact that they are only rarely updated. Yet, this may decrease the spectral gap of the graph and slow convergence down.

Proof. The proof consists in evaluating \( \|v_t^{i+1} - x^*\|_{\lambda_+ A}^2 \) and follows the same scheme as by Nesterov and Stich (2017). However, \( F_A^* \) is not strongly convex because matrix \( A^T A \) is generally not full rank. Yet, \( F_A^* \) is strongly convex for the pseudo-norm \( \lambda^+ A \) and the value of \( F_A^*(x) \) only depends on the value of \( x \) on \( \text{Ker}(A)^\perp \). Gower et al. (2018) develop a similar proof in the quadratic case but without assuming any specific structure on \( A \). The detailed proof can be found in Appendix C.

### 3.3 Practical algorithm

Algorithm 1 is written in a form that is convenient for analysis but it is not practical at all. Its logically equivalent implementation is described in Algorithm 2. All nodes run the same procedure with a different rank \( r \) and their own local functions \( f_r \) and variables \( \theta_r, v_t(r) \) and \( y_t(r) \). For convenience, we define \( B = \begin{pmatrix} 1 - \theta & \theta \\ \delta & 1 - \delta \end{pmatrix} \) and \( s_{ij} = (\theta p_i^2 / \sigma_i, \sigma_\text{inv} \mu_{ij}^2 \mid \theta) \).

Note that each update only involves two nodes, thus allowing for many updates to be run in parallel. Algorithm 2 is obtained by multiplying the updates of Algorithm 1 by \( A \) on the left. This has the benefit of switching from edge variables (of size \( E \times d \)) to node variables (of size \( n \times d \)). Then, if \( y_t \) corresponds to the variable of Algorithm 1, \( y_t(i) = e^T y \) represents the local \( y_t \) variable of node \( i \) and is used to compute the gradient of \( f_i^* \). We obtain \( v_t(i) \) in the same way. The updates can be expressed as a matrix multiplication (contraction step, making \( y_t \) and \( v_t \) closer), plus a gradient term which is equal to 0 if the node is not at one end of the sampled edge. The multiplication by \( B^{t \rightarrow r} \) corresponds to catching up the global contraction steps for updates in which node \( r \) did not take part. The form of \( s_{ij} \) comes from the fact that \( A e_i e_j^T A = \mu_{ij}^2 (e_i - e_j)(e_i - e_j)^T \).

### Algorithm 2 Asynchronous Decentralized Accelerated Coordinate Descent

1. \( r \{ \text{Id of the node} \} \)
2. \( \text{seed} \{ \text{The common seed} \} \)
3. \( z_0 = 0, y_0(r) = 0, v_0(r) = 0, t = 0 \)
4. Initialize random generator with \( \text{seed} \)
5. **while** \( t < T \) **do**
6. Sample \( e \) from \( P \)
7. if \( \exists j / e \in \{(r, j), (j, r)\} \) then
8. \( \begin{align*}
    v_t(r)^T &= B^{t \rightarrow r} v_t(r)^T \\
    y_t(r)^T &= y_t(r)^T \\
    z_t &= \nabla f^*_r (y_t(r)) \\
    \text{send\_gradient} (x_t, j) &\{ \text{non blocking} \}
\end{align*} \)
9. \( z_{\text{dist}} = \text{receive\_gradient} (j) \{ \text{blocking} \} \)
10. \( g_t(r) = s_t (z_t - z_{\text{dist}}) \)
11. \( \begin{align*}
    v_{t+1}^r(r)^T &= B (v_t(r)^T - g_t(r)^T) \\
    y_{t+1}(r)^T &= y_t(r)^T \\
    t_r &= t + 1
\end{align*} \)
12. **end if**
13. \( t = t + 1 \)
14. **end while**
15. **return** \( z_r \)

### 3.4 Communication schedule

Even though updates are actually local, nodes need to keep track of the total number of updates performed (variable \( t \)) in order to properly execute Algorithm 2.

This problem can be handled by generating in advance the sequence of all communications and then simply unrolling this sequence as the algorithm progresses. All nodes perform the neighbors selection protocol starting with the same seed and only consider the communications they are involved in. Therefore, they can count the number of iterations completed.

This way of selecting neighbours can cause some nodes to wait for the gradient of a busy node before they can actually perform their update. Since the communication schedule is defined in advance, they cannot choose a free neighbor and exchange with him instead. However, any way of making edges sampled independent from the previous ones would be equivalent to generating the sequence in advance. Indeed, choosing free neighbors over busy ones would introduce correlations with the current state and therefore with the edges sampled in the past.

### 4 Performances

#### 4.1 Homogeneous decentralized networks

In this section, we introduce two network-related assumptions under which the performances of ES-DACD are provably comparable to the performances...
of randomized gossip averaging or SSDA. We denote $p_{\text{max}} = \max_{ij} p_{ij}$ and $p_{\text{min}} = \min_{ij} p_{ij}$. We also note $\bar{p}(G) = \max_{i} p_i$ and $\underline{p}(G) = \min_{i} p_i$ the maximum and minimum probabilities of nodes of a graph $G$ where $p_i = \sum_{j=1}^n p_{ij}$. We note $d_{\text{max}}$ and $d_{\text{min}}$ the maximum and minimum degrees in the graph. The dependence on $G$ will be omitted when clear from the context.

**Assumption 1.** We say that a family of graph $G$ with edge weights $p$ is quasi-regular if there exists a constant $c$ such that for $n \in \mathbb{N}$, $p_{\text{max}} \leq cp_{\text{min}}$ and $d_{\text{max}} \leq cd_{\text{min}}$.

Assumption 1 is satisfied for many standard graphs and probability distribution over edges. In particular, it is satisfied by the uniform distribution for regular degree graphs.

**Assumption 2.** The family of graphs $G$ is such that there exists a constant $c$ such that for $n \in \mathbb{N}$, $\max_{ij} e_{ij}^T A^+ A e_{ij} \leq c_n^2$ where $A$ is of the form of Equation (9) with $\mu_{ij} = 1$ and uniquely defines $G(n)$.

This second assumption essentially means that removing one edge or another should have a similar impact on the connectivity of the graph. It is verified with $c = 1$ if the graph is completely symmetric (ring or complete graph). Since $A^+ A$ is a projector, $e_{ij}^T A^+ A e_{ij} \leq 1$ so Assumption 2 holds true any time the ratio $\frac{e_{ij}^T A^+ A e_{ij}}{\mu_{ij}}$ is bounded below. In particular, the grid, the hypercube, or any random graph with bounded degree respect Assumption 2.

### 4.2 Average time per iteration

ESDACD updates are much cheaper than the updates of any global synchronous algorithm such as SSA. However, the partial synchrony discussed in Section 3.4 may drastically slow the algorithm down, making it inefficient to use cheaper iterations. Theorem 2 shows that this does not happen for regular graphs with homogeneous probabilities. We note $\tau_{\text{max}}$ the maximum delay of all edges.

**Theorem 2.** If we denote $T_{\text{max}}(k)$ the time taken by ESDACD to perform $k$ iterations when edges are sampled according to the distribution $p$:

$$\bar{\tau} = \mathbb{E}\left[\frac{1}{k} T_{\text{max}}(k)\right] \leq c \bar{p}\tau_{\text{max}}$$

with a constant $c < 14$.

The proof of Theorem 2 is in Appendix A. Note that the constant can be improved in some settings, for example if all nodes have the same degrees and all edges have the same weight then a tighter bound $c < 4$ holds.

**Corollary 1.** If $G$ satisfies Assumption 1 then there exists $c > 0$ such that for any $n \in \mathbb{N}$, the expected average time per iteration taken by ESDACD in $G(n)$ when edges are sampled uniformly verifies:

$$\mathbb{E}[T_{\text{max}}(k)] \leq c \frac{\tau_{\text{max}}}{n} k + o(k).$$

Corollary 1 shows that when all nodes have comparable activation frequencies then the expected time required to complete one ESDACD iteration scales as the inverse of the number of nodes in the network. This result essentially means that the synchronization cost of locking edges does not grow with the size of the network and so iterations will not be longer on a bigger network. At any given time, a constant fraction of the nodes is actively performing an update (rather than waiting for a message) and this fraction does not shrink as the network grows. The time per iteration can be as high as $\tau_{\text{max}}$ for some graph topologies that break Assumption 1, e.g., star networks. These topologies are more suited to centralized algorithms because some nodes take part in almost all updates.

### 4.3 Distributed average consensus

Algorithm 2 solves the problem of distributed gossip averaging if we set $f_i(\theta) = \frac{1}{2} \|\theta - c_i\|^2$. In this setting, $f_i^k(x) = \frac{1}{2} \|x + c_i\|^2 - \frac{1}{2} \|c_i\|^2$ so $\nabla f_i^k(x) = x + c_i$. Local smoothness and strong convexity parameters are all equal to 1.

At each round, an edge is chosen and nodes exchange their current estimate of the mean (which is equal to $e_{ij}^T y_i + c_i$ for node $i$). Yet, they do not update it directly but they keep two sequences $y_i$ and $v_i$ that are updated according to a linear system. One step simply consists in doing a convex combination of these values at the previous step, plus a mixing of the current value with the value of the chosen neighbor.

The standard randomized gossip iteration consists in choosing an edge $(i, j)$ and replacing the current values of nodes $i$ and $j$ by their average. If we denote $E_2(t)$ the second moment of the error at time $t$:

$$E_2(t) \leq (1 - \theta_{\text{gossip}})^{2t} E_2(0),$$

where $\theta_{\text{gossip}} = \lambda_{\text{min}}^+(\bar{W})$, with $\bar{W} = \frac{1}{F} L$ if $L$ is the Laplacian matrix of the graph (Boyd et al., 2006).

**Corollary 2.** If $G$ satisfies Assumption 2 then there exists $c > 0$ such that for any $n \in \mathbb{N}$, if $\theta_{\text{ESDACD}}$ is the rate ESDACD in $G(n)$ and $\theta_{\text{gossip}}$ the rate of randomized gossip averaging when edges are sampled uniformly then they verify:

$$\theta_{\text{ESDACD}} \geq \frac{c}{\sqrt{n}} \sqrt{\theta_{\text{gossip}}}.$$
Actually, we can show that in this case, ESDACD iterations are exactly the same as randomized gossip iterations. In the case of the ring graph, $\theta_{\text{gossip}} \approx n^{-3}$ and so $\theta_{\text{ESDACD}} \approx n^{-2}$ which is significantly better for $n$ large. For the grid graph, a similar analysis yields $\theta_{\text{ESDACD}} = O(n^{-3/2})$. Achieving this message complexity on a grid is an active research area and is often achieved with complex algorithms like geographic gossip (Dimakis et al., 2006), relying on overlay networks, or LADA (Li et al., 2007), using lifted Markov chains (Diaconis et al., 2000). Although synchronous gossip algorithms achieved this rate (Oreshkin et al., 2010), finding an asynchronous algorithm that could match the rates of geographic gossip was still, to the best of our knowledge, an open area of research (Dimakis et al., 2010).

Therefore, ESDACD shows improved rate compared with standard gossip when the eigengap of the gossip matrix is small. To our knowledge, this is the first time that better convergence rates of the second moment of the error are proven. Indeed, though they both show improved rates in expectation, the shift-register approach (Cao et al., 2006; Liu et al., 2013) has no proven rates for the second moment and the rates for the second moment of heavy ball gossip (Loizou and Richtárik, 2018) do not improve over standard randomized gossip averaging. Surprisingly, our results show that gossip averaging is best analyzed as a special case of a more general optimization algorithm that is not even restricted to quadratic objectives. Standard acceleration techniques shed a new light on the problem and allows for a better understanding of it.

We acknowledge that the improved rates of convergence do not come for free. The accelerated gossip algorithm requires some global knowledge on the graph (eigenvalues of the gossip matrix and probability of activating each edge). Even though these quantities can be approximated relatively well for simple graphs with a known structure, evaluating them can be more challenging for more complex graphs (and can be even harder than or of equivalent difficulty to the problem of average consensus). Yet, we believe that ESDACD as a gossip algorithm can still be practical in many cases, in particular when values need to be averaged over the same network multiple times or when computing resources are available at some time but not at the time of averaging. Such use cases can typically be encountered in sensor networks, in which the computation of such hyperparameters can be anticipated before deployment. In any case, the analysis shows that standard optimization tools are useful to analyze randomized gossip algorithms.

### 4.4 Comparison to SSDA

The results described in Theorem 1 are rather precise and allow for a fine tuning of the edges probabilities depending on the topology of the graph and of the local smoothnesses. However, the rate cannot always be expressed in a way that makes it simple to compare with SSDA.

**Corollary 3.** Let $G$ be a family of graph verifying Assumptions 1 and 2. There exists $c > 0$ such that:

$$\frac{\theta_{\text{ESDACD}}}{\tau_{\text{ESDACD}}} \geq c \frac{1}{\tau_{\text{max}}} \sqrt{\frac{\gamma}{\kappa}} = c \frac{\theta_{\text{SSDA}}}{\tau_{\text{SSDA}}}$$

where $\theta_{\text{ESDACD}}$ is the rate of ESDACD when edges are sampled uniformly and $\theta_{\text{SSDA}}$ the rate of SSDA when both algorithms use matrix $A$ as defined in Equation (9).

The proof is in Appendix B. Actually, sampling does not need to be uniform but a ratio $\sqrt{p_{\text{min}}/p_{\text{max}}}$ would appear in the constant otherwise. The result of Corollary 3 means that asynchrony comes almost for free for decentralized gradient descent in these cases. Indeed, both algorithms scale similarly in the network and optimization parameters. Note that in this case, we compare ESDACD and SSDA (and not MSDA) meaning that we implicitly assume that communication times are greater than computing times. This is because ESDACD is very efficient in terms of communication but not necessarily in terms of gradients.

Corollary 3 states that the rates per unit of time are similar. Figure 1 compares the two algorithms in terms of network and computational resources usage. SSDA iterations require all nodes to send messages to all their neighbors, resulting in a very high communication cost. ESDACD avoids this cost by only performing local updates. SSDA uses $n/2$ times more gradients per iteration so both algorithms have a comparable cost in terms of gradients.

At each SSDA iteration, nodes need to wait for the slowest node in the system whereas many nodes can be updated in parallel with ESDACD. ESDACD can thus be tuned not to sample slow edges too much, or on the opposite to sample quick edges but with highly non-smooth nodes at both ends more often.

Edge updates yield a strong correlation between the probabilities of sampling edges and the final rate. In heterogeneous cases (in terms of functions to optimize as well as network characteristics), the greater flexibility of ESDACD allows for a better fine-tuning of the parameters (step-size) and thus for better rates.
Choosing because we chose constant and simple learning rates. Gossip. ESDACD is slightly slower at the beginning and heavy ball \( \omega = \beta = 5 \) for heavy ball gossip \((\omega = 0, \beta = 0)\) for standard and heavy ball gossip \((\omega = 0, \beta = 0)\) for standard. We chose the best performing parameters from the original paper \((\omega = 1, \beta = 0.5)\) for heavy ball gossip. ESDACD is slightly slower at the beginning because we chose constant and simple learning rates. Choosing \( B_0 \) and \( A_0 \) from Appendix C as in Nesterov and Stich (2017) would lead to a more complex algorithm with better initial performances.

5.2 ESDACD vs. SSDA

In order to assess the performances of the algorithm in a fully controlled setting, we perform experiments on two synthetic datasets, similar to the one used by Scaman et al. (2017):

- Regression: Each node \( i \) has a vector of \( N \) observations, noted \( X_i \in \mathbb{R}^{d \times N} \) with \( d = 50 \) drawn from a centered Gaussian with variance 1. The targets \( y_{i,j} \) are obtained by applying function \( g : x \rightarrow \bar{x}_{i,j} + \cos(x_{i,j}) + \epsilon \) where \( \bar{x}_{i,j} = d^{-1} 1^T X_i e_j \) and \( \epsilon \) is a centered Gaussian noise with variance 0.25. At each node, the loss function is \( f_i(\theta_i) = \frac{1}{2} \| X_i^T \theta - y_i \|^2 + c_i \| \theta \|^2 \) with \( c_i = 1 \).

- Classification: Each node \( i \) has a vector of \( N \) observations, noted \( X_i \in \mathbb{R}^{d \times N} \) with \( d = 50 \). Observations are drawn from a Gaussian of variance 1 centered at \(-1\) for the first class and \(1\) for the second class. Classes are balanced. At each node, the loss function is \( f_i(\theta_i) = \sum_{j=1}^N \ln \left( 1 + \exp^{-y_{i,j} X_i^T \theta} \right) + c_i \| \theta \|^2 \) with \( c_i = 1 \).

Our main focus is on the speed of execution. Recall that edge \((i,j)\) takes time \( \tau_{ij} \) to transmit a message and so if node \( i \) starts its \( k_i \)th update at time \( t_i(k_i) \) then \( t_i(k_i + 1) = \max_{l \leq i,j} t_l(k_i) + \tau_{ij} \) and the same for \( j \). This gives a simple recursion to compute the time needed to execute the algorithm in an idealized setting, that we use as the x-axis for the plots.

To perform the experiments, the gossip matrix chosen for SSDA is the Laplacian matrix and \( \mu_j^2 = p_j^2 (\sigma_j^{-1} + \sigma^{-1})^{-1} \) is chosen for ESDACD. The error plotted is the maximum suboptimality \( \max_i F(\theta_i) - \min_x F(x) \).

5 Experiments

5.1 ESDACD vs. gossip averaging

The goal of this part is to illustrate the rate difference depending on the topology of the graph. We study graphs of \( n \) nodes where 10% of the nodes have value 1 and the rest have value 0. Similar results are obtained with values drawn from Gaussian distributions.

Figures 2 and 3 show that ESDACD consistently beats standard and heavy ball gossip (Loizou and Richtárik, 2018). The clear rates difference for the ring graph shown in Figure 3 illustrates the fact that ESDACD scales far better for graphs with low connectivity. We chose the best performing parameters from the original paper \((\omega = 1, \beta = 0.5)\) for heavy ball gossip. ESDACD is slightly slower at the beginning because we chose constant and simple learning rates. Choosing \( B_0 \) and \( A_0 \) from Appendix C as in Nesterov and Stich (2017) would lead to a more complex algorithm with better initial performances.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Improvement</th>
<th>Communications</th>
<th>Gradients computed</th>
<th>Speed</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSDA</td>
<td>( \sqrt{\frac{n}{\kappa_i}} )</td>
<td>2E</td>
<td>n</td>
<td>( O\left(\frac{1}{n}\right))</td>
</tr>
<tr>
<td>ESDACD</td>
<td>( O\left(\frac{1}{n^{\frac{1}{2}}}\right))</td>
<td>2</td>
<td>2</td>
<td>( O\left(\frac{1}{n}\right))</td>
</tr>
</tbody>
</table>

Figure 1: Per iteration costs of SSDA and ESDACD for quasi-regular graphs.

Figure 2: ESDACD, pairwise gossip and heavy ball gossip on the 10 × 10 grid.

Figure 3: ESDACD, pairwise gossip and heavy ball gossip on the ring graph of size 100.
Experiments are conducted on the $10 \times 10$ grid network. We perform $n/4$ times more iteration for ESDACD than for SSDA. Therefore, in our experiments, an execution of SSDA uses roughly 2 times more gradients and 8 times more messages (for the grid graph) than an execution of ESDACD. This also allows us to compare the resources used by the 2 algorithms.

**Homogeneous setting:** In this setting, we choose uniform constant delays and $N = 150$ for each node. We notice on Figure 4 that SSDA is roughly two times faster than ESDACD, meaning that $n/8$ ESDACD iterations are completed in parallel by the time SSDA completes one iteration. This means that in average, a quarter of the nodes are actually waiting to complete the schedule, since 2 nodes engage in each iteration.

**Heterogeneous setting:** In this setting, $N$ is uniformly sampled between 50 (problem dimension) and 300, thus leading to very different values for the local condition numbers. Delays are all exponentially distributed with parameter 1. Figure 5 shows that ESDACD is computationally more efficient than SSDA on the regression problem because it has a far lower final error although it uses 2 times less gradients. This can be explained by larger step sizes along regular edges and suggests that ESDACD adapts more easily to changes in local regularity, even with uniform sampling probabilities. ESDACD is also much faster since in average, each node performs 2 iterations in half the time needed for one SSDA iteration. For the classification problem, strong convexity is more homogeneous because it only comes from regularization. Therefore, ESDACD does not take full advantage of the local structure of the problem and show performances that are similar to those of SSDA.

**6 Conclusion**

In this paper, we introduced the **Edge Synchronous Dual Accelerated Coordinate Descent** (ESDACD), a randomized gossip algorithm for the optimization of sums of smooth and strongly convex functions. We showed that it matches the performances of SSDA, its synchronous counterpart. Empirically, ESDACD even outperforms SSDA in heterogeneous settings. Applying ESDACD to the distributed average consensus problem yields the first asynchronous gossip algorithm that provably achieves better rates in variance than the standard randomized gossip algorithm, for example matching the rate of geographic gossip (Dimakis et al., 2006) on a grid.

Promising lines of work include a communication accelerated version that would match the speed of MSDA (Scaman et al., 2017) when computations are more expensive than communications, a fully asynchronous extension that could handle late gradients as well as a stochastic version of the algorithm that would only use stochastic gradients of the local functions.
Acknowledgement

We acknowledge support from the European Research Council (grant SEQUOIA 724063).

References


Ji Liu and Stephen J Wright. Asynchronous stochastic coordinate descent: Parallelism and convergence
Accelerated Decentralized Optimization with Local Updates for Smooth and Strongly Convex Objectives


