Accelerated decentralized optimization with local updates for smooth and strongly convex objectives

Hadrien Hendrikx, Francis Bach, Laurent Massoulié

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 κ 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 $O((\tau_{\max} + \Delta_{\max})\sqrt{\kappa/\gamma}\ln(\epsilon^{-1}))$ to reach a precision ϵ where γ is the spectral gap of the graph, $\tau_{\max}$ the maximum communication delay and $\Delta_{\max}$ the maximum computation time. Therefore, it matches the rate of SSDA (Scaman et al., 2017), which is optimal when $\tau_{\max} = \Omega(\Delta_{\max})$. Applying ESDACD to quadratic local functions leads to an accelerated randomized gossip algorithm of rate $O(\sqrt{\theta_{\text{gossip}}/n})$ where θ_{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

Proceedings of the 22[nd] International Conference on Artificial Intelligence and Statistics (AISTATS) 2019, Naha, Okinawa, Japan. PMLR: Volume 89.

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 $\mathrm{Ker}(W) = \mathrm{Vec}(\mathbb{1})$ 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 $\lambda^+_{\min}(W) \leq \frac{1}{n-1}$ 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 σ_i-strongly convex, which means that for all $x, y \in \mathbb{R}^d$:

$$f_i(x) - f_i(y) \leq \nabla f_i(y)^T(x-y) + \frac{L_i}{2}\|x-y\|^2 \quad (1)$$

$$f_i(x) - f_i(y) \geq \nabla f_i(y)^T(x-y) + \frac{\sigma_i}{2}\|x-y\|^2. \quad (2)$$

Note that the fenchel conjugate f_i^* of f_i (defined in Equation (8)) is (L_i^{-1})-strongly convex and (σ_i^{-1})-smooth, as shown in Kakade et al. (2009). We denote $L_{\max} = \max_i L_i$ and $\sigma_{\min} = \min_i \sigma_i$. Then, we denote $\kappa_l = \frac{L_{\max}}{\sigma_{\min}}$. κ_l 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:

$$\min_{x \in \mathbb{R}^{n\times d}:\ x_i = x_j\ \forall i,j \in \{1,\dots,n\}} F(x). \quad (3)$$

We assume that a communication between nodes $i, j \in V$ takes a time τ_{ij}. If $(i,j) \notin E$, the communication is impossible so $\tau_{ij} = \infty$. Node i takes time Δ_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 needed to obtain 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 $\text{Ker}(A^T) = \text{Vec}(\mathbb{1})$ where $\mathbb{1} = \sum_{i=1}^n e_i$ 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:

$$\min_{x\in\mathbb{R}^{n\times d}:\ A^T x=0} F(x). \tag{4}$$

This problem is equivalent to the following one:

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

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

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

because F is convex and $A^T\mathbb{1} = 0$. Then, we obtain the dual formulation of this problem, which writes:

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

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 Ae_{ij} 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 $e_{ij}^T A^T \mathbb{1} = 0$ for all $(i,j) \in E$ and $\text{Ker}(A^T) \subset \text{Vec}(\mathbb{1})$ 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 μ_{ij} are chosen to be equal to 1. When not explicitly stated, all μ_{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 $\|x\|_{A^+A}^2 = x^T A^+ A x$ 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

$$\theta^2 = \min_{ij} \frac{p_{ij}^2}{\mu_{ij}^2 e_{ij}^T A^+ A e_{ij}} \frac{\sigma_A}{\sigma_i^{-1} + \sigma_j^{-1}} \geq \frac{\sigma_A}{S^2}. \tag{10}$$

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

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

Algorithm 1 Asynchronous Decentralized Accelerated Coordinate Descent

$y_0 = 0$, $v_0 = 0$, $t = 0$
while $t < T$ **do**
  Sample (i,j) with probability p_{ij}
  $y_{t+1} = (1-\delta)y_t + \delta v_t - \eta_{ij} g_{ij}(y_t)$
  $v_{t+1} = (1-\theta)v_t + \theta y_t - \frac{\theta}{\sigma_A p_{ij}} g_{ij}(y_t)$
end while

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

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

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

Theorem 1 shows that Algorithm 1 converges with rate θ. 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 $L_{\max}\left(\sigma_i^{-1}+\sigma_j^{-1}\right)$ term whereas the other terms are strictly related to the topology of the graph. Parameter θ is invariant to the scale of μ because rescaling μ would also multiply $\lambda^+_{\min}(A^TA)$ 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 μ_{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 $\|v_{t+1}-x^*\|^2_{A^+A}$ and follows the same scheme as by Nesterov and Stich (2017). However, F_A^* is not strongly convex because matrix A^TA is generally not full rank. Yet, F_A^* is strongly convex for the pseudo-norm A^+A and the value of $F_A^*(x)$ only depends on the value of x on $\mathrm{Ker}(A)^\perp$. 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 θ_r, $v_t(r)$ and $y_t(r)$. For convenience, we define $B=\begin{pmatrix}1-\theta & \theta\\ \delta & 1-\delta\end{pmatrix}$ and $s_{ij}=\begin{pmatrix}\frac{\theta\mu_{ij}^2}{p_{ij}\sigma_A} & \mu_{ij}^2\eta_{ij}\end{pmatrix}^T$.

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, $y_t(i)=e_i^TAy_t$ 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-t_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 $Ae_{ij}e_{ij}^TA^T=\mu_{ij}^2(e_i-e_j)(e_i-e_j)^T$.

Algorithm 2 Asynchronous Decentralized Accelerated Coordinate Descent

r {Id of the node}
seed {The common seed}
$z_r=0$, $y_0(r)=0$, $v_0(r)=0$, $t=0$
Initialize random generator with *seed*
while $t<T$ **do**
  Sample e from P
  if $\exists j$ / $e\in\{(r,j),(j,r)\}$ **then**
    $\begin{pmatrix}v_t(r)^T\\ y_t(r)^T\end{pmatrix}_r = B^{t-t_r}\begin{pmatrix}v_{t_r}(r)^T\\ y_{t_r}(r)^T\end{pmatrix}$
    $z_r=\nabla f_r^*\left(y_t(r)\right)$
    $send\_gradient(x_r,j)$ {*non blocking*}
    $z_{dist}=receive\_gradient(j)$ {*blocking*}
    $g_t(r)=s_e\left(z_r-z_{dist}\right)$
    $\begin{pmatrix}v_{t+1}(r)^T\\ y_{t+1}(r)^T\end{pmatrix}_r = B\begin{pmatrix}v_t(r)^T\\ y_t(r)^T\end{pmatrix}-g_t(r)^T$
    $t_r=t+1$
  end if
  t = t + 1
end while
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 ESDACD are provably comparable to the performances

of randomized gossip averaging or SSDA. We denote $p_{\max} = \max_{ij} p_{ij}$ and $p_{\min} = \min_{ij} p_{ij}$. We also note $\bar{p}(\mathcal{G}) = \max_i p_i$ and $\underline{p}(\mathcal{G}) = \min_i p_i$ the maximum and minimum probabilities of nodes of a graph $\mathcal{G}$ where $p_i = \sum_{i=1}^n p_{ij}$. We note $d_{\max}$ and $d_{\min}$ the maximum and minimum degrees in the graph. The dependence on $\mathcal{G}$ will be omitted when clear from the context.

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

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

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 $\frac{n}{E}$ 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 SSDA. 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 $\tau_{\max}$ the maximum delay of all edges.

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

$$\bar{\tau} = \mathbb{E}\left[\frac{1}{k} T_{\max}(k)\right] \leq c\bar{p}\tau_{\max} \tag{14}$$

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 $\mathcal{G}$ 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 $\mathcal{G}(n)$ when edges are sampled uniformly verifies:*

$$\mathbb{E}\left[T_{\max}(k)\right] \leq c\frac{\tau_{\max}}{n} k + o(k). \tag{15}$$

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 $\tau_{\max}$ 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, $f_i^*(x) = \frac{1}{2}\|x + c_i\|^2 - \frac{1}{2}\|c_i\|^2$ and so $\nabla f_i^*(x) = x + c_i$. 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 $e_i^T y_t + c_i$ for node i). Yet, they do not update it directly but they keep two sequences y_t and v_t 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 $\mathcal{E}_2(t)$ the second moment of the error at time t:

$$\mathcal{E}_2(t) \leq (1 - \theta_{\text{gossip}})^{2t} \mathcal{E}_2(0), \tag{16}$$

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

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

$$\theta_{ESDACD} \geq \frac{c}{\sqrt{n}}\sqrt{\theta_{\text{gossip}}}. \tag{17}$$

We can use tools from Mohar (1997) to estimate the eigenvalues of usual graphs. In the case of the complete graph, $\theta_{\text{gossip}} \approx n^{-1}$ and so $\theta_{ESDACD} \approx \theta_{\text{gossip}}$.

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 $\theta_{ESDACD} \approx n^{-2}$ which is significantly better for n large. For the grid graph, a similar analysis yields $\theta_{ESDACD} = O(n^{-3/2})$. 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 $\mathcal{G}$ be a family of graph verifying Assumptions 1 and 2. There exists $c > 0$ such that:*

$$\frac{\theta_{ESDACD}}{\bar{\tau}_{ESDACD}} \geq c \frac{1}{\tau_{\max}} \sqrt{\frac{\gamma}{\kappa}} = c \, \frac{\theta_{SSDA}}{\bar{\tau}_{SSDA}}, \tag{18}$$

where θ_{ESDACD} is the rate of ESDACD when edges are sampled uniformly and θ_{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 $\sqrt{p_{\min}/p_{\max}}$ 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 iterations 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.

Algorithm	Improvement	Communications	Gradients computed	Speed
SSDA	$\sqrt{\frac{\gamma}{\kappa_l}}$	2E	n	1
ESDACD	$O\left(\frac{1}{n}\sqrt{\frac{\gamma}{\kappa_l}}\right)$	2	2	$O\left(\frac{1}{n}\right)$

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

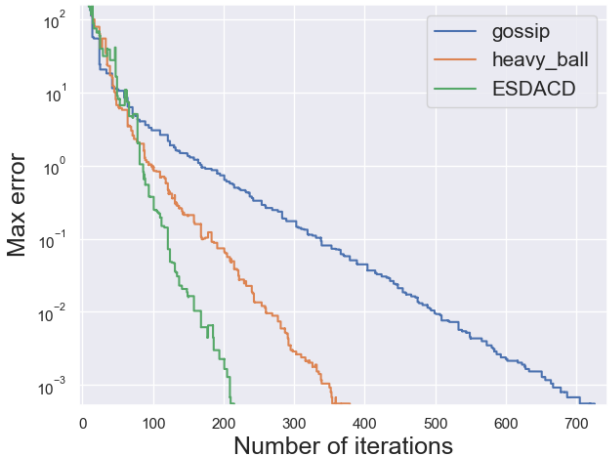

Figure 2: ESDACD, pairwise gossip and heavy ball gossip on the 10×10 grid.

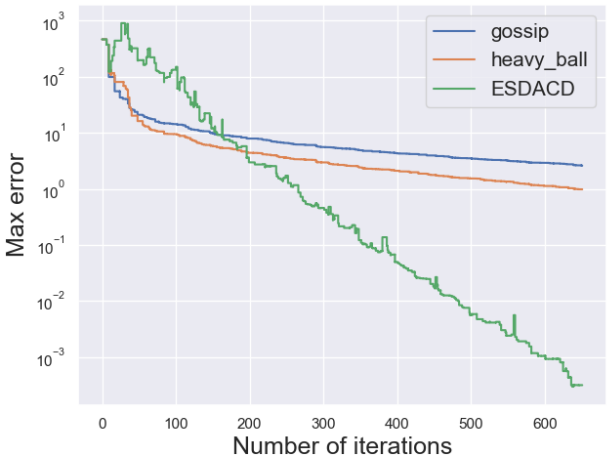

Figure 3: ESDACD, pairwise gossip and heavy ball gossip on the ring graph of size 100.

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 ($\omega = 1$ and $\beta = 0.5$) 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 $g : x \to \bar{x}_{i,j} + cos(\bar{x}_j) + \epsilon$ where $\bar{x}_j = d^{-1}\mathbb{1}^T X_i e_j$ and ϵ 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 $f_i(\theta_i) = \sum_{j=1}^{N} \ln\left(1 + \exp^{-y_{i,j}X_{i,j}^T\theta}\right) + c_i\|\theta\|^2$ with $c_i = 1$.

Our main focus is on the speed of execution. Recall that edge (i, j) takes time τ_{ij} to transmit a message and so if node i starts its k_ith update at time $t_i(k_i)$ then $t_i(k_i + 1) = \max_{l=i,j} t_l(k_l) + \tau_{ij}$ 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 $\mu_{ij}^2 = p_{ij}^2(\sigma_i^{-1} + \sigma_j^{-1})^{-1}$ is chosen for ESDACD. The error plotted is the maximum suboptimality $\max_i F(\theta_i) - \min_x F(x)$.

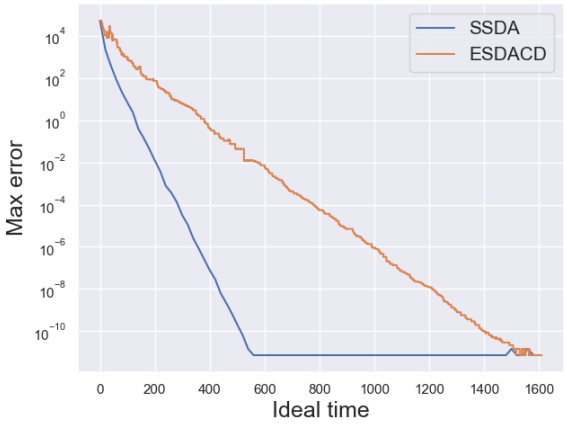

Figure 4: Homogeneous regression problem

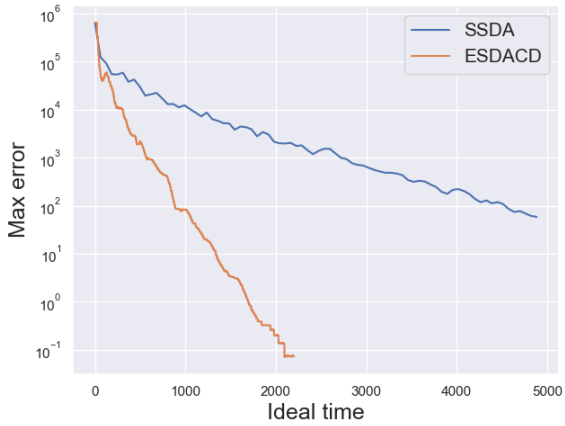

Figure 5: Heterogeneous regression problem.

Experiments are conducted on the 10×10 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 SSDAis 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

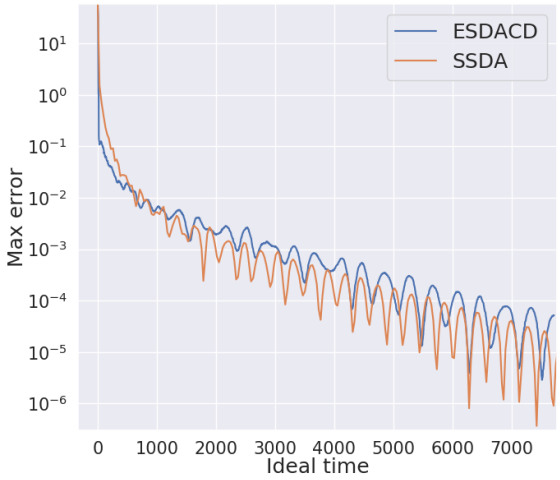

Figure 6: Heterogeneous classification problem.

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

Zeyuan Allen-Zhu, Zheng Qu, Peter Richtárik, and Yang Yuan. Even faster accelerated coordinate descent using non-uniform sampling. In *International Conference on Machine Learning*, pages 1110–1119, 2016.

Richard Arratia and Louis Gordon. Tutorial on large deviations for the binomial distribution. *Bulletin of mathematical biology*, 51(1):125–131, 1989.

François Baccelli, Guy Cohen, Geert Jan Olsder, and Jean-Pierre Quadrat. *Synchronization and linearity: an algebra for discrete event systems*. John Wiley & Sons Ltd, 1992.

Stephen Boyd, Arpita Ghosh, Balaji Prabhakar, and Devavrat Shah. Randomized gossip algorithms. *IEEE transactions on information theory*, 52(6): 2508–2530, 2006.

Stephen Boyd, Neal Parikh, Eric Chu, Borja Peleato, Jonathan Eckstein, et al. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends® in Machine learning*, 3(1):1–122, 2011.

Ming Cao, Daniel A Spielman, and Edmund M Yeh. Accelerated gossip algorithms for distributed computation. In *Proc. of the 44th Annual Allerton Conference on Communication, Control, and Computation*, pages 952–959. Citeseer, 2006.

Samprit Chatterjee and Eugene Seneta. Towards consensus: Some convergence theorems on repeated averaging. *Journal of Applied Probability*, 14(1):89–97, 1977.

Igor Colin, Aurélien Bellet, Joseph Salmon, and Stéphan Clémençon. Gossip dual averaging for decentralized optimization of pairwise functions. *arXiv preprint arXiv:1606.02421*, 2016.

Morris H DeGroot. Reaching a consensus. *Journal of the American Statistical Association*, 69(345):118–121, 1974.

Persi Diaconis, Susan Holmes, and Radford M Neal. Analysis of a nonreversible markov chain sampler. *Annals of Applied Probability*, pages 726–752, 2000.

Ralf Diekmann, Andreas Frommer, and Burkhard Monien. Efficient schemes for nearest neighbor load balancing. *Parallel computing*, 25(7):789–812, 1999.

Alexandros G Dimakis, Anand D Sarwate, and Martin J Wainwright. Geographic gossip: efficient aggregation for sensor networks. In *Proceedings of the 5th international conference on Information processing in sensor networks*, pages 69–76. ACM, 2006.

Alexandros G Dimakis, Soummya Kar, José MF Moura, Michael G Rabbat, and Anna Scaglione. Gossip algorithms for distributed signal processing. *Proceedings of the IEEE*, 98(11):1847–1864, 2010.

John C Duchi, Alekh Agarwal, and Martin J Wainwright. Dual averaging for distributed optimization: Convergence analysis and network scaling. *IEEE Transactions on Automatic control*, 57(3):592–606, 2012.

Olivier Fercoq and Peter Richtárik. Accelerated, parallel, and proximal coordinate descent. *SIAM Journal on Optimization*, 25(4):1997–2023, 2015.

Euhanna Ghadimi, Iman Shames, and Mikael Johansson. Multi-step gradient methods for networked optimization. *IEEE Trans. Signal Processing*, 61(21): 5417–5429, 2013.

Robert M Gower, Filip Hanzely, Peter Richtárik, and Sebastian Stich. Accelerated stochastic matrix inversion: general theory and speeding up bfgs rules for faster second-order optimization. *arXiv preprint arXiv:1802.04079*, 2018.

Robert Mansel Gower and Peter Richtárik. Stochastic dual ascent for solving linear systems. *arXiv preprint arXiv:1512.06890*, 2015.

Robert Hannah, Fei Feng, and Wotao Yin. `A2BCD`: An asynchronous accelerated block coordinate descent algorithm with optimal complexity. *arXiv preprint arXiv:1803.05578*, 2018.

Sham Kakade, Shai Shalev-Shwartz, and Ambuj Tewari. On the duality of strong convexity and strong smoothness: Learning applications and matrix regularization. *Unpublished Manuscript, http://ttic. uchicago. edu/shai/papers/KakadeShalevTewari09. pdf*, 2:1, 2009.

Wenjun Li, Huaiyu Dai, and Y Zhang. Location-aided fast distributed consensus. *IEEE Transactions on Information Theory*, 2007.

Xiangru Lian, Ce Zhang, Huan Zhang, Cho-Jui Hsieh, Wei Zhang, and Ji Liu. Can decentralized algorithms outperform centralized algorithms? a case study for decentralized parallel stochastic gradient descent. In *Advances in Neural Information Processing Systems*, pages 5330–5340, 2017a.

Xiangru Lian, Wei Zhang, Ce Zhang, and Ji Liu. Asynchronous decentralized parallel stochastic gradient descent. *arXiv preprint arXiv:1710.06952*, 2017b.

Ji Liu and Stephen J Wright. Asynchronous stochastic coordinate descent: Parallelism and convergence

properties. *SIAM Journal on Optimization*, 25(1): 351–376, 2015.

Ji Liu, Brian DO Anderson, Ming Cao, and A Stephen Morse. Analysis of accelerated gossip algorithms. *Automatica*, 49(4):873–883, 2013.

Ji Liu, Stephen J Wright, Christopher Ré, Victor Bittorf, and Srikrishna Sridhar. An asynchronous parallel stochastic coordinate descent algorithm. *The Journal of Machine Learning Research*, 16(1):285–322, 2015.

Nicolas Loizou and Peter Richtárik. Accelerated gossip via stochastic heavy ball method. In *Allerton*, 2018.

Bojan Mohar. Some applications of laplace eigenvalues of graphs. In *Graph symmetry*, pages 225–275. Springer, 1997.

Aryan Mokhtari and Alejandro Ribeiro. Dsa: Decentralized double stochastic averaging gradient algorithm. *The Journal of Machine Learning Research*, 17(1):2165–2199, 2016.

Ion Necoara, Yurii Nesterov, and François Glineur. Random block coordinate descent methods for linearly constrained optimization over networks. *Journal of Optimization Theory and Applications*, 173 (1):227–254, 2017.

Angelia Nedic and Asuman Ozdaglar. Distributed subgradient methods for multi-agent optimization. *IEEE Transactions on Automatic Control*, 54(1):48–61, 2009.

Angelia Nedic, Alex Olshevsky, and Wei Shi. Achieving geometric convergence for distributed optimization over time-varying graphs. *SIAM Journal on Optimization*, 27(4):2597–2633, 2017.

Yurii Nesterov. Efficiency of coordinate descent methods on huge-scale optimization problems. *SIAM Journal on Optimization*, 22(2):341–362, 2012.

Yurii Nesterov. *Introductory lectures on convex optimization: A basic course*, volume 87. Springer Science & Business Media, 2013.

Yurii Nesterov and Sebastian U Stich. Efficiency of the accelerated coordinate descent method on structured optimization problems. *SIAM Journal on Optimization*, 27(1):110–123, 2017.

Boris N Oreshkin, Mark J Coates, and Michael G Rabbat. Optimization and analysis of distributed averaging with short node memory. *IEEE Transactions on Signal Processing*, 58(5):2850–2865, 2010.

S Sundhar Ram, A Nedić, and Venugopal V Veeravalli. Asynchronous gossip algorithms for stochastic optimization. In *Decision and Control, 2009 held jointly with the 2009 28th Chinese Control Conference. CDC/CCC 2009. Proceedings of the 48th IEEE Conference on*, pages 3581–3586. IEEE, 2009.

S Sundhar Ram, Angelia Nedić, and Venugopal V Veeravalli. Distributed stochastic subgradient projection algorithms for convex optimization. *Journal of optimization theory and applications*, 147(3):516–545, 2010.

Benjamin Recht, Christopher Re, Stephen Wright, and Feng Niu. Hogwild: A lock-free approach to parallelizing stochastic gradient descent. In *Advances in neural information processing systems*, pages 693–701, 2011.

Peter Richtárik and Martin Takáč. Parallel coordinate descent methods for big data optimization. *Mathematical Programming*, 156(1-2):433–484, 2016.

Kevin Scaman, Francis Bach, Sébastien Bubeck, Yin Tat Lee, and Laurent Massoulié. Optimal algorithms for smooth and strongly convex distributed optimization in networks. In *International Conference on Machine Learning*, pages 3027–3036, 2017.

Wei Shi, Qing Ling, Gang Wu, and Wotao Yin. Extra: An exact first-order algorithm for decentralized consensus optimization. *SIAM Journal on Optimization*, 25(2):944–966, 2015.

Hanlin Tang, Xiangru Lian, Ming Yan, Ce Zhang, and Ji Liu. d^2: Decentralized training over decentralized data. *arXiv preprint arXiv:1803.07068*, 2018.

Lin Xiao, Stephen Boyd, and Sanjay Lall. A scheme for robust distributed sensor fusion based on average consensus. In *Information Processing in Sensor Networks, 2005. IPSN 2005. Fourth International Symposium on*, pages 63–70. IEEE, 2005.

Martin Zinkevich, Markus Weimer, Lihong Li, and Alex J Smola. Parallelized stochastic gradient descent. In *Advances in neural information processing systems*, pages 2595–2603, 2010.

A Detailed average time per iteration proof

The goal of this section is to prove Theorem 2. The proof develops an argument similar to the one of Theorem 8.33 (Baccelli et al., 1992). Yet, the theorem cannot be used directly and we need to specialize the argument for our problem in order to get a tighter bound. We note t the number of iterations that the algorithm performs, and we introduce the random variable $X^t(i,w)$ such that if edge (i,j) is activated at time $t+1$ (with probability p_{ij}), then for all $w \in \mathbb{N}^*$:

$$X^{t+1}(i,w) = X^t(i,w-1) + X^t(j,w-1).$$

and $X^{t+1}(k,w-1) = X^t(k,w-1)$ otherwise. We start with the initial conditions $X^0(i,0) = 1$ and $X^0(i,w) = 0$ for any $w > 0$. The following lemma establishes a relationship between the time taken by the algorithm to complete t iterations and variables X^t.

Lemma 1. *If we note $T_{\max}(t)$ the time at which the last node of the system finishes iteration t then for all $\theta > 0$:*

$$\mathbb{E}\left[T_{\max}(t)\right] \leq \theta t + \sum_{w \geq \theta t} \sum_{i=1}^{n} \mathbb{E}\left[X^t(i,w)\right].$$

Proof. We first prove by induction on t that if we denote $T_i(t)$ the time at which node i finishes iteration t, then for any $i \in \{1,..,n\}$:

$$T_i(t) = \max_{w \in \mathbb{N}, X^t(i,w) > 0} w. \tag{19}$$

To ease notations, we write $w_{\max}(i,t) = \max_{w \in \mathbb{N}, X^t(i,w)>0} w$. The property is true for $t = 0$ because $T_i(0) = 0$ for all i.

We now assume that it is true for some fixed $t > 0$ and we assume that edge (k,l) has been activated at time t. For all $i \notin \{k,l\}$, $T_i(t+1) = T_i(t)$ and for all $w \in \mathbb{N}^*$, $X^{t+1}(i,w-1) = X^t(i,w-1)$ so the property is true. Besides,

$$\begin{aligned} w_{\max}(k,t+1) &= \max_{w \in \mathbb{N}^*, X^t(k,w-1)+X^t(l,w-1)>0} w \\ &= \max_{w \in \mathbb{N}, X^t(i,w)+X^t(i,w)>0} w+1 \\ &= 1 + \max\left(w_{\max}(k,t), w_{\max}(l,t)\right) \\ &= 1 + \max\left(T_k(t), T_l(t)\right) = T_k(t+1). \end{aligned}$$

We finish the proof of Equation (19) by observing that k and l are completely equivalent.

The form of the recurrence guarantees that for any fixed $t \in \mathbb{N}$ and $w > 1$, if there exists i such that $X^t(i,w) > 0$ then for any $w' < w$, there exists j such that $X^t(j,w') > 0$. Therefore,

$$T_{\max}(t) = \max_i \max_{w \in \mathbb{N}, X^t(i,w)>0} w = \max_{w \in \mathbb{N}, \sum_i X^t(i,w)>0} w = \sum_{w \in \mathbb{N}} \mathbb{1}\left(\sum_{i=1}^{n} X^t(i,w) \geq 1\right), \tag{20}$$

because having $X^t(i,w) > 0$ is equivalent to having $X^t(i,w) \geq 1$ since $X^t(i,w)$ is integer valued. Therefore, for any $\theta \in [0,1]$

$$T_{\max}(t) \leq \theta t + \sum_{w \geq \theta t} \mathbb{1}\left(\sum_{i=1}^{n} X^t(i,w) \geq 1\right),$$

and the proof results from taking the expectation of the previous inequality and using Markov inequality on the second term. □

Although there is still a maximum in the expression of $T_i(t)$, the recursion for variable X has a much simpler form. In particular, we will crucially exploit its linearity. We write $p_i = \sum_j p_{ij}$ and introduce $\underline{p} = \min_i p_i$ and $\bar{p} = \max_i p_i$. We now prove the following Lemma:

Lemma 2. *For all i, if $\delta_1 = \underline{p}$, $\delta_2 = \bar{p}$ and $\delta = \frac{2\delta_2 - \delta_1}{1-2\delta_2}$ then for all $\theta > 0$*

$$\sum_{p \geq \theta t} \mathbb{E}\left[X^t(i,p)\right] \leq (1+\delta)^t \, \mathbb{P}\left[Binom(2\delta_2, t) \geq \theta t\right]. \tag{21}$$

Proof. Taking the expectation over the edges that can be activated gives:

$$\mathbb{E}\left[X^{t+1}(i,w)\right] = (1-p_i)\,\mathbb{E}\left[X^t(i,w)\right] + \sum_j p_{ij}\mathbb{E}\left[X^t(j,w-1)\right] + p_i\mathbb{E}\left[X^t(i,w-1)\right]. \tag{22}$$

In particular, for all i, $\mathbb{E}\left[X^{t+1}(i,w)\right] \leq \bar{X}^t(w)$ where $\bar{X}^0(w) = 1$ if $w = 0$ and 0 otherwise, and:

$$\bar{X}^{t+1}(w) = (1-\underline{p})\,\bar{X}^t(w) + 2\bar{p}\bar{X}^t(w-1). \tag{23}$$

We now introduce $\phi^t(z) = \sum_{w\in\mathbb{N}} z^w \bar{X}^t(w)$. A direct recursion leads to:

$$\phi^t(z) = (1 - \underline{p} + 2\bar{p}z)^t. \tag{24}$$

Then, using the fact that $\delta > 0$:

$$\phi_t(z) \leq (1+\delta)^t \left(1 - 2\delta_2 + \frac{2\delta_2}{1+\delta}z\right)^t \leq (1+\delta)^t \left(1 - 2\delta_2 + 2\delta_2 z\right)^t = (1+\delta)^t \phi_{bin}(2\delta_2, t)(z), \tag{25}$$

where $\phi_{bin}(2\delta_2, t)$ is the generating function of the Binomial law of parameters $2\delta_2$ and t. The inequalities above on the integral series ϕ_t and $(1+\delta)^t\phi_{bin}(2\delta_2,t)$ actually hold coefficient by coefficient. Therefore, $\mathbb{E}\left[X^t(i,p)\right] \leq (1+\delta)^t \mathbb{P}\left(Binom(2\delta_2, t) = p\right)$ □

We conclude the proof of the theorem with this last lemma:

Lemma 3. *If $\theta = 6\delta_2 + \delta$ then:*

$$\lim_{t\in\mathbb{N}} \sum_{w \geq \theta t} \mathbb{E}\left[X^t(i,w)\right] = 0 \tag{26}$$

Proof. We use tail bounds for the Binomial distribution (Arratia and Gordon, 1989) in order to get for $\theta \geq 2\delta_2$:

$$\ln \mathbb{P}\left[Binom(2\delta_2, t) \geq \theta t\right] \leq -tD(\theta||2\delta_2), \tag{27}$$

where $D(p||q) = p\ln\frac{p}{q} + (1-p)\ln\frac{1-p}{1-q}$ so applying Lemma 2 yields:

$$\sum_{w \geq \theta t} \mathbb{E}\left[X^t(i,w)\right] \leq e^{-t[D(\theta||2\delta_2) - \ln(1+\delta)]}. \tag{28}$$

Therefore, we are left to prove that $D(\theta||2\delta_2) - \ln(1+\delta) > 0$. However,

$$D(\theta||2\delta_2) = 2\delta_2 \ln(\frac{2\delta_2}{\theta}) + (1-2\delta_2)\ln\frac{1-2\delta_2}{1-\theta} \geq 2\delta_2\ln(\frac{2\delta_2}{\theta}) - 2\delta_2 + \theta \tag{29}$$

by using that $\frac{x}{1+x} \leq \ln(1+x) \leq x$. Since $\theta = 6\delta_2 + \delta$ and $\delta \leq \frac{2\delta_2}{1-2\delta_2}$, assuming that $\delta_2 \leq \frac{3}{8}$ yields:

$$D(\theta||2\delta_2) \geq 2\delta_2 \left[2 - \ln(3 + \frac{\delta}{2\delta_2})\right] + \delta > \delta \geq \ln(1+\delta). \tag{30}$$

If $\delta_2 \geq \frac{3}{8}$, then $\theta > 1$ so the result is obvious because $X^t(i,w) = 0$ for $w > t$.

□

B Execution speed comparisons

B.1 Comparison with gossip

In this section, we prove Corollary 2.

Proof. We consider a matrix A such that $Ae_{ij} = \mu_{ij}(e_i - e_j)$ and $\mu_{ij}^2 = \frac{1}{2}$ for all $(i,j) \in E$. Then multiplying by $W_{ij} = Ae_{ij}e_{ij}^T A^T$ corresponds to averaging the values of nodes i and j and so the rate of uniform randomized gossip averaging depends on $\bar{W} = \mathbb{E}[W_{ij}]$.

In this case, applying ESDACD with matrix A yields a rate of

$$\theta_{ESDACD} = \min_{ij} \frac{p_{ij}}{\mu_{ij}\sqrt{\sigma_i^{-1} + \sigma_j^{-1}}} \frac{\sqrt{\lambda_{\min}^+(A^TA)}}{\sqrt{e_{ij}A^+Ae_{ij}}} \geq \sqrt{\frac{\lambda_{\min}^+(AA^T)}{cnE}} \tag{31}$$

where c is a constant independent of the size of the graph coming from Assumption 2.

Since $\bar{W} = \frac{1}{E}AA^T$ then $\theta_{\text{gossip}} = \frac{1}{E}\lambda_{\min}^+(AA^T)$ and so:

$$\theta_{ESDACD} \geq \frac{c'}{\sqrt{n}}\sqrt{\theta_{\text{gossip}}} \tag{32}$$

with $c' = c^{-\frac{1}{2}}$. □

B.2 Comparison with SSDA

In this section, we prove Corollary 3. SSDA is based on an arbitrary gossip matrix whereas the rate of ESDACD is based on a specific matrix A^TA where $Ae_{ij} = \mu_{ij}(e_i - e_j)$. Yet, $W = AA^T$ is a perfectly valid gossip matrix. Indeed, $\text{Ker}(W) = \text{Ker}(A) = Vec(\mathbb{1})$ and AA^T is an $n \times n$ symmetric positive matrix defined on the graph $\mathcal{G}(n)$. Besides, $\lambda_{\min}^+(A^TA) = \lambda_{\min}^+(AA^T)$, which enables us to compare the rates of SSDA and ESDACD.

Proof. For arbitrary μ, the rate of ESDACD writes:

$$\theta_{ESDACD} \geq \min_{ij} \frac{p_{ij}}{\mu_{ij}\sqrt{L_{\max}(\sigma_i^{-1} + \sigma_j^{-1})e_{ij}^TA^+Ae_{ij}}}\sqrt{\lambda_{\min}^+(A^TA)}. \tag{33}$$

Here, we choose $\mu_{ij}^2 = \frac{1}{2}$, which yields the bound:

$$\theta_{ESDACD} \geq p_{\min}\sqrt{\frac{\lambda_{\max}(AA^T)}{\max_{ij} e_{ij}^TA^+Ae_{ij}}}\sqrt{\frac{\gamma}{\kappa}}. \tag{34}$$

Therefore, combining this with Theorem 2 and Assumption 2 gives:

$$\frac{\theta_{ESDACD}}{\bar{\tau}_{ESDACD}} \geq \frac{p_{\min}\sqrt{E}}{c\bar{p}\tau_{\max}}\sqrt{\frac{\lambda_{\max}(AA^T)}{n}}\sqrt{\frac{\gamma}{\kappa}} \geq \frac{c'}{\tau_{\max}}\frac{p_{\min}}{p_{\max}}\sqrt{\frac{d_{\min}}{d_{\max}}}\sqrt{\frac{E}{nd_{\max}}}\sqrt{\frac{\gamma}{\kappa}} \tag{35}$$

where we have used that $\lambda_{\max} \geq \frac{1}{n} Tr(AA^T) \geq d_{\min}$ and $\bar{p} \leq p_{\max} d_{\max}$. We then use Assumption 1 to get that there exists c'' such that:

$$\frac{\theta_{ESDACD}}{\bar{\tau}_{ESDACD}} \geq \frac{c''}{\tau_{\max}} \sqrt{\frac{\gamma}{\kappa}} = c'' \frac{\theta_{SSDA}}{\bar{\tau}_{SSDA}} \tag{36}$$

□

In the proof above, it appears that having probabilities that are too unbalanced harms the convergence rate of ESDACD. However, if these probabilities are carefully selected to match the square root of the smoothness along the edge, and if delays are such that this does not cause very slow edges to be sampled too often then unbalanced probabilities can greatly boost the convergence rate.

C Detailed rate proof

The proof of Theorem 1 is detailed in this section. Recall that we note A^+ the pseudo-inverse of A and we define the scalar product $\langle x, y \rangle_{A^+A} = x^T A^+ A y$. The associated norm is a semi-norm because A^+A is positive semi-definite. Since A^+A is a projector on the orthogonal of $\mathrm{Ker}(A)$, it is a norm on the orthogonal of $\mathrm{Ker}(A)$.

Our proof follows the key steps of Nesterov and Stich (2017). However, we study the problem in the norm defined by A^+A because our problem is strongly convex only on the orthogonal of $\mathrm{Ker}(A)$. Matrix A can be tuned so that F_A^* has the same smoothness in all directions, thus leading to optimal rates. We start by two small lemmas to introduce the strong convexity and smoothness inequalities for the A^+A semi-norm. We note $U_{ij} = e_{ij} e_{ij}^T$.

Lemma 4 (Strong convexity of F_A^*)**.** *For all $x, y \in \mathbb{R}^E$,*

$$F_A^*(x) - F_A^*(y) \geq \nabla F_A^*(y)^T (x - y) + \frac{\sigma_A}{2} \|x - y\|_{A^+A}^2 \tag{37}$$

with $\sigma_A = \frac{\lambda_{\min}^+(A^T A)}{L_{\max}}$

Proof. Inequality (37) is obtained by writing the strong convexity inequality for each f_i^* and then summing them. Then, we remark that $L_i \leq L_{\max}$ for all i and that $\|Aw\|^2 = \|Aw\|_{A^+A}^2 \geq \lambda_{\min}^+(A^T A) \|w\|_{A^+A}^2$ for $w = x - y$. More specifically:

$$\begin{aligned}
F_A^*(x) - F_A^*(y) &= \sum_{i=1}^n \left(f_i^* \left(e_i^T A x \right) - f_i^* \left(e_i^T A y \right) \right) \\
&\geq \sum_{i=1}^n \nabla f_i^* \left(e_i^T A y \right)^T e_i^T (Ax - Ay) + \frac{1}{2} (Ax - Ay)^T \left(\sum_{i=1}^n L_i^{-1} e_i e_i^T \right) (Ax - Ay) \\
&\geq \nabla F_A^*(y)^T (x - y) + \frac{1}{2 L_{\max}} (x - y)^T A^T A (x - y) \\
&\geq \nabla F_A^*(y)^T (x - y) + \frac{\lambda_{\min}(A^T A)}{2 L_{\max}} \|x - y\|_{A^+A}^2
\end{aligned}$$

□

Lemma 5 (Smoothness of F_A^*)**.** *We note $x_{t+1} = y_t - h_{kl} U_{kl} \nabla F_A^*(y_t)$ where $h_{kl}^{-1} = \mu_{kl}^2 (\sigma_k^{-1} + \sigma_l^{-1})$. If edge (k, l) is sampled at time t,*

$$F_A^*(x_{t+1}) - F_A^*(y_t) \leq -\frac{1}{2 \mu_{kl}^2 \left(\sigma_k^{-1} + \sigma_l^{-1} \right)} \|U_{kl} \nabla F_A^*(y_t)\|^2. \tag{38}$$

Equation (38) can be seen as an *ESO* inequality (Richtárik and Takáč, 2016) applied to the directional update $h_{kl} U_{kl} \nabla F_A^*(y_t)$.

Proof. Assuming that edge (k,l) is drawn at time t, we use that each f_i^* is (σ_i^{-1})-smooth to write:

$$f_i^*\left(e_i^T A x_{t+1}\right) - f_i^*\left(e_i^T A y_t\right) \leq -h_{kl}\nabla f_i^*\left(e_i^T A y_t\right)^T e_i^T A U_{kl}\nabla F_A^*(y_t) + \frac{1}{2\sigma_i}\|h_{kl}e_i^T A U_{kl}\nabla F_A^*(y_t)\|^2.$$

Summing it over all values of i gives:

$$F_A^*(x_{t+1}) - F_A^*(y_t) \leq \nabla F_A^*(y_t)^T\left[-h_{kl}U_{kl} + \frac{1}{2}h_{kl}^2 U_{kl}A^T\sum_{i=1}^n \sigma_i^{-1}e_i e_i^T A U_{kl}\right]\nabla F_A^*(y_t).$$

Then, we decompose by using that $Ae_{ij} = \mu_{ij}(e_i - e_j)$ and $U_{kl} = e_{kl}e_{kl}^T$ to get that

$$F_A^*(x_{t+1}) - F_A^*(y_t) \leq \nabla F_A^*(y_t)^T U_{kl}\left[-h_{kl} + \frac{1}{2}h_{kl}^2\mu_{kl}^2(\sigma_k^{-1}+\sigma_l^{-1})\right]\nabla F_A^*(y_t).$$

We conclude the proof by using the fact that $h_{kl} = \frac{1}{\mu_{kl}^2(\sigma_k^{-1}+\sigma_l^{-1})}$. □

We can now start the proof of Theorem 1. We first prove the convergence of a different algorithm which is essentially the one by Nesterov and Stich (2017) and show that Algorithm 1 is obtained for a specific choice of initial conditions.

Proof. More specifically, we choose $A_0, B_0 \in \mathbb{R}$ and recursively define the following coefficients:

$$a_{t+1}^2 S^2 = A_{t+1}B_{t+1} \tag{39}$$
$$B_{t+1} = B_t + \sigma_A a_{t+1} \tag{40}$$
$$A_{t+1} = A_t + a_{t+1} \tag{41}$$
$$\alpha_t = \frac{a_{t+1}}{A_{t+1}} \tag{42}$$
$$\beta_t = \frac{\sigma_A a_{t+1}}{B_{t+1}}. \tag{43}$$

Then, we take arbitrary $x_0, y_0, v_0 \in \mathbb{R}^{E\times d}$ and recursively define:

$$y_t = \frac{(1-\alpha_t)x_t + \alpha_t(1-\beta_t)v_t}{1-\alpha_t\beta_t} \tag{44}$$
$$v_{t+1} = (1-\beta_t)v_t + \beta_t y_t - \frac{a_{t+1}}{B_{t+1}p_{ij}}U_{ij}\nabla F_A^*(y_t) \tag{45}$$
$$x_{t+1} = y_t - \frac{1}{\mu_{ij}^2(\sigma_i^{-1}+\sigma_j^{-1})}U_{ij}\nabla F_A^*(y_t). \tag{46}$$

For convenience, we write $w_t = (1-\beta_t)v_t + \beta_t y_t$. Then, we study the quantity $r_t^2 = \|v_t - x^*\|_{A^+A}^2$ where x^* is the minimizer of F_A^*. Recall that $g_{ij}(y_t) = \frac{a_{t+1}}{B_{t+1}p_{ij}}U_{ij}\nabla F_A^*(y_t)$.

$$\|v_{t+1} - x^*\|_{A^+A}^2 = \|w_t - x^*\|_{A^+A}^2 + \|\frac{a_{t+1}}{B_{t+1}p_{ij}}U_{ij}\nabla F_A^*(y_t)\|_{A^+A}^2 - 2\frac{a_{t+1}}{B_{t+1}p_{ij}}\nabla F_A^*(y_t)^T U_{ij}A^+A(w_t - x^*). \tag{47}$$

Then,

$$\mathbb{E}_{ij}[\frac{a_{t+1}}{B_{t+1}p_{ij}}\nabla F_A^*(y_t)^T U_{ij}] = \sum_{ij} p_{ij}\frac{a_{t+1}}{B_{t+1}p_{ij}}\nabla F_A^*(y_t)^T U_{ij} = \frac{a_{t+1}}{B_{t+1}}\nabla F_A^*(y_t)^T. \tag{48}$$

Therefore, Equation (47) can be rewritten:

$$\mathbb{E}\left[r_{t+1}^2\right] \leq \mathbb{E}\left[\|w_t - x^*\|_{A^+A}^2\right] + \mathbb{E}\left[\frac{e_{ij}^T A^+ A e_{ij} a_{t+1}^2}{B_{t+1}^2 p_{ij}^2}\|U_{ij}\nabla F_A^*(y_t)\|^2\right] - 2\frac{a_{t+1}}{B_{t+1}}\nabla F_A^*(y_t)^T(w_t - x^*). \tag{49}$$

Now, the goal is to write a smoothness equation to control the middle term and make $F_A^*(x_{t+1})$ appear. This control is provided by Equation (38) in Lemma 5.

Therefore, if we choose S such that for all (i,j), $\frac{e_{ij}^T A^+ A e_{ij}(\sigma_i^{-1}+\sigma_j^{-1})\mu_{ij}^2}{p_{ij}^2} \leq S^2$ then the equation becomes:

$$\|v_{t+1} - x^*\|_{A^+A}^2 \leq \|w_t - x^*\|_{A^+A}^2 + \frac{2S^2 a_{t+1}^2}{B_{t+1}^2}\left[F_A^*(y_t) - \mathbb{E}\left[F_A^*(x_{t+1})\right]\right] - 2\frac{a_{t+1}}{B_{t+1}}\nabla F_A^*(y_t)^T(w_t - x^*). \tag{50}$$

We use the convexity of the squared norm to get that $\|w_t - x^*\|_{A^+A}^2 \leq (1-\beta_t)r_t^2 + \beta_t\|y_t - x^*\|_{A^+A}^2$. Then, if we multiply both sides by B_{t+1} we get:

$$B_{t+1}r_{t+1}^2 \leq B_t r_t^2 + \beta_t B_{t+1}\|y_t - x^*\|_{A^+A}^2 + \frac{2S^2 a_{t+1}^2}{B_{t+1}}\left[F_A^*(y_t) - \mathbb{E}\left[F_A^*(x_{t+1})\right]\right] - 2a_{t+1}\nabla F_A^*(y_t)^T(w_t - x^*). \tag{51}$$

We can now use Equation (37) of Lemma 4 (strong convexity of F_A^* in norm A^+A) to write that:

$$\begin{aligned}
-a_{t+1}\nabla F_A^*(y_t)^T(w_t - x^*) &= a_{t+1}\nabla F_A^*(y_t)^T A^+A\left(x^* - y_t + \frac{1-\alpha_t}{\alpha_t}(x_t - y_t)\right) \\
&\leq a_{t+1}\left(F_A^*(x^*) - F_A^*(y_t) - \frac{1}{2}\sigma_A\|y_t - x^*\|_{A^+A}^2 + \frac{1-\alpha_t}{\alpha_t}(F_A^*(x_t) - F_A^*(y_t))\right) \\
&\leq a_{t+1}F_A^*(x^*) - A_{t+1}F_A^*(y_t) + A_t F_A^*(x_t) - \frac{1}{2}a_{t+1}\sigma_A\|y_t - x^*\|_{A^+A}^2.
\end{aligned}$$

Then, we combine the previous inequality with Equation (51) and we use the fact that $B_{t+1}\beta_t = a_{t+1}\sigma_A$ so that:

$$B_{t+1}r_{t+1}^2 \leq B_t r_t^2 + 2A_{t+1}\left[F_A^*(y_t) - \mathbb{E}\left[F_A^*(x_{t+1})\right]\right] - 2\left[(A_{t+1} - A_t)F_A^*(x^*) - A_{t+1}F_A^*(y_t) + A_t F_A^*(x_t)\right], \tag{52}$$

and so:

$$B_{t+1}r_{t+1}^2 - B_t r_t^2 \leq 2A_t\left[F_A^*(x_t) - F_A^*(x^*)\right] - 2A_{t+1}\left[\mathbb{E}\left[F_A^*(x_{t+1})\right] - F_A^*(x^*)\right]. \tag{53}$$

By summing over all inequalities, we get that

$$2A_t\mathbb{E}\left[F_A^*(x_t) - F_A^*(x^*)\right] + B_t\mathbb{E}[r_t^2] \leq r_0^2. \tag{54}$$

Now, we need to estimate the growth of coefficients A_t and B_t. We prove by induction on t that if $A_0 = 1$ and $B_0 = \sigma_A$ then for all $t \in \mathbb{N}$, $\alpha_t = \beta_t = \frac{\sqrt{\sigma_A}}{S}$ $A_t = \left(1 - \frac{\sqrt{\sigma_A}}{S}\right)^{-t}$ and $B_t = \sigma_A A_t$.

We can first combine Equation (41) and Equation (42) to obtain

$$a_{t+1}(\alpha_t^{-1} - 1) = A_t \tag{55}$$

$$a_{t+1}(\beta_t^{-1} - 1) = \frac{B_t}{\sigma_A} \tag{56}$$

For $t = 0$, we can combine equations (55) and (56) to obtain that $\alpha_0^{-1} - 1 = \beta_0^{-1} - 1$ (since $a_1 \neq 0$ and so $\alpha_0 = \beta_0$. Finally,

$$a_1^2 S^2 = A_1 B_1 = \frac{a_1^2 \sigma_A}{\alpha_0 \beta_0}$$

and so $\alpha_0 = \beta_0 = \frac{\sqrt{\sigma_A}}{S}$.

Now suppose that the property is true for a given $t \geq 0$. Then, we use Equation (55) and the fact that $A_{t+1} = a_{t+1} + A_t$. Since $1 + (\alpha_t^{-1} - 1)^{-1} = \frac{\alpha_t^{-1} - 1 + 1}{\alpha_t^{-1} - 1} = (1 - \alpha_t)^{-1}$ then by induction assumption, $A_{t+1} = \left(1 - \frac{\sqrt{\sigma_A}}{S}\right)^{-t-1}$.

We use Equation (56) in the same way to prove that $B_{t+1} = \sigma_A A_{t+1}$.

Then, we use equations (55) and (56) at time $t+1$ to get that $\alpha_{t+1}^{-1} - 1 = \beta_{t+1}^{-1} - 1$ so $\alpha_{t+1} = \beta_{t+1}$. Their value can again be retrieved by using Equation (39), which finishes the induction.

We have proven that for this choice of A_0 and B_0 the α and β coefficients are constant and are equal to $\theta = \frac{\sqrt{\sigma_A}}{S}$. Therefore, $v_{t+1} = (1 - \theta) v_t + \theta y_t - \frac{\theta}{p_{ij} \sigma_A} U_{ij} \nabla F_A^*(y_t)$. With this choice of parameters, y_{t+1} can be expressed as:

$$y_{t+1} = \frac{(1-\theta) x_{t+1} + \theta(1-\theta) v_{t+1}}{1 - \theta^2} = \frac{x_{t+1} + \theta v_{t+1}}{1 + \theta}.$$

Then, the coefficients of Algorithm 1 are recovered by replacing x_{t+1} and v_{t+1} by their expressions in Equations (46) and (45). The actual values of a_{t+1}, A_{t+1} and B_{t+1} are only used for the analysis because only $\frac{a_{t+1}}{B_{t+1}} = \frac{\sigma_A}{\beta_t}$ appears in the recursion. □