DADAO: Decoupled Accelerated Decentralized Asynchronous Optimization

Adel Nabli [1 2] **Edouard Oyallon** [1]

Abstract

This work introduces DADAO: the first decentralized, accelerated, asynchronous, primal, first-order algorithm to minimize a sum of L-smooth and μ-strongly convex functions distributed over a given network of size n. Our key insight is based on modeling the local gradient updates and gossip communication procedures with separate independent Poisson Point Processes. This allows us to decouple the computation and communication steps, which can be run in parallel, while making the whole approach completely asynchronous. This leads to communication acceleration compared to synchronous approaches. Our new method employs primal gradients and does not use a multi-consensus inner loop nor other ad-hoc mechanisms such as Error Feedback, Gradient Tracking, or a Proximal operator. By relating the inverse of the smallest positive eigenvalue of the Laplacian matrix χ_1 and the maximal resistance $\chi_2 \leq \chi_1$ of the graph to a sufficient minimal communication rate between the nodes of the network, we show that our algorithm requires $\mathcal{O}(n\sqrt{\frac{L}{\mu}}\log(\frac{1}{\epsilon}))$ local gradients and only $\mathcal{O}(n\sqrt{\chi_1\chi_2}\sqrt{\frac{L}{\mu}}\log(\frac{1}{\epsilon}))$ communications to reach a precision ϵ, up to logarithmic terms. Thus, we simultaneously obtain an accelerated rate for both computations and communications, leading to an improvement over state-of-the-art works, our simulations further validating the strength of our relatively unconstrained method.

1. Introduction

In recent years, the increased amount of available data as well as the proliferation of highly-parallelizable and connected hardware have brought significant changes in the way we process data. These developments have led to the need for efficient and scalable methods for distributed optimization, particularly in the context of machine learning. Indeed, in scenarios where data is distributed across multiple nodes, such as in edge computing or distributed sensor networks, leveraging the local resources of each device is a topic of significant interest. In other settings, such as clusters, spreading the compute load is ideally done to obtain a linear speedup in the number of nodes. In a typical distributed training framework, the goal is to minimize a sum of functions $(f_i)_{i\leq n}$ split across n nodes of a computer network. A corresponding optimization procedure involves alternating local computations on the nodes and communications along the edges $\mathcal{E}$ of the network. In the decentralized setting, there is no central machine aggregating the information sent by the workers: nodes are only allowed to communicate with their neighbors in the network. This work addresses simultaneously multiple limitations of existing decentralized algorithms while guaranteeing fast convergence rates.

Synchronous lock. Optimal methods (Scaman et al., 2017; Kovalev et al., 2021a) have been derived for synchronous first-order algorithms, whose executions are blocked until all nodes have reached a predefined state (*e.g.*, they must all finish computing local gradients before the round of communication begins), which limits their efficiency in practice as they can heavily be impacted by a few slow nodes or edges in the graph (the *straggler problem*). To tackle the synchronous lock, we rely on the continuized framework (Even et al., 2021a), itself derived from the randomized gossip model (Boyd et al., 2006). In randomized gossip, nodes update their local values at random times using pairwise communication updates named gossip. Thus, iterates are randomized, labeled with a continuous-time index that only need local clocks to be synchronized at the beginning of the procedure (in opposition to a *global* iteration count that has to be known by all at all time) and performed locally with no regards to a specific global ordering of events. While based on discrete events and thus readily implementable, the continuized framework simplifies the analysis by leveraging continuous proof tools.

[1]Sorbonne Université, CNRS, ISIR, Paris, France [2]Mila, Concordia University, Montréal, Canada. Correspondence to: Adel Nabli <adel.nabli@sorbonne-universite.fr>.

Proceedings of the 40th International Conference on Machine Learning, Honolulu, Hawaii, USA. PMLR 202, 2023. Copyright 2023 by the author(s).

Coupled lock. However, in (Even et al., 2021a), gradient and gossip operations are coupled: each communication along an edge first requires the computation of the gradients of the two functions locally stored on the corresponding nodes. As more communication steps than gradient computations are necessary to reach an ϵ precision, even in an optimal framework (Kovalev et al., 2021a; Scaman et al., 2017), the coupling leads to an overload in terms of gradient steps. Moreover, coupling computations and communications implies they must be performed *sequentially*, decoupling them allows both tasks to be done in *parallel*, allowing an additional speedup.

To our knowledge, our work is the first primal method to tackle those locks simultaneously while obtaining accelerated rates for both computations and communications. We propose a novel algorithm (DADAO: Decoupled Accelerated Decentralized Asynchronous Optimization) based on a combination of similar formulations to (Kovalev et al., 2021a; Even et al., 2021b; Hendrikx, 2022) in the continuized framework of (Even et al., 2021a). We study:

$$\inf_{x\in\mathbb{R}^d} \sum_{i=1}^{n} f_i(x)\,, \tag{1}$$

where each $f_i : \mathbb{R}^d \to \mathbb{R}$ is a μ-strongly convex and L-smooth function computed in one of the n nodes of a network. We derive a first-order optimization algorithm that only uses primal gradients and relies on Point-wise Poisson Processes (P.P.P.s (Last & Penrose, 2017)) modeling of the communication and gradient occurrences, leading to accelerated communication and computation rates. Furthermore, our communication bounds rely on the maximal resistance of a graph rather than the largest eigenvalue of a Laplacian, leading to an additional acceleration compared to works which rely on synchrony. Our framework is based on a simple fixed-point iteration and kept minimal: it only involves primal computations with an additional momentum term. Thus, we do not add other cumbersome designs such as the Error-Feedback or Forward-Backward used in (Kovalev et al., 2021a) (whose adaptation to asynchronous settings is for now unclear). While we do not consider the delays bound to appear in practice (we assume instantaneous communications and computations), we remove the coupling lock by performing gradient and gossip steps in parallel. Tab. 1 compares DADAO with other approaches and shows it is the only work to achieve accelerated rates both in number of communication and gradients.

Contributions. **(1)** We propose a primal algorithm with provable guarantees in the context of asynchronous decentralized learning. **(2)** This algorithm is the first to reach accelerated rates for both communications and computations while not requiring ad-hoc mechanisms obtained from an inner loop. **(3)** We propose a simple theoretical framework compared to concurrent works, we show that our rates are better than previous works, and **(4)** we illustrate this theoretical comparison numerically.

Structure of the paper. In Sec. 3.1, we describe our work hypothesis and our model of a decentralized environment, while Sec. 3.2 describes our dynamic. Sec. 3.3 states our convergence guarantees and highlights that the communication and computational rates of our method are better compared to its competitors. Next, Sec. 4.1 explains our implementation of this algorithm, and finally, Sec. 4.2 verifies our claims numerically. All our experiments are reproducible, using PyTorch (Paszke et al., 2019), our code being online [1].

Notations: $f = \mathcal{O}(g)$ means there is a constant $C > 0$ such that $|f| \leq C|g|$, $\{e_i\}_{i\leq d}$ is the canonical basis of $\mathbb{R}^d, d \in \mathbb{N}$, $\mathbf{1}$ is the vector of 1, $\mathbf{I}$ the identity, A^+ is the pseudo-inverse of A. We further write $\mathbf{e}_i \triangleq e_i \otimes \mathbf{I}$.

2. Related Work

Continuized and asynchronous algorithms. We highly rely on the elegant continuized framework (Even et al., 2021a), which allows obtaining simpler proofs and brings the flexibility of asynchronous algorithms. We reemphasize that identically to (Even et al., 2021a), the result of this paper is *a stochastic discrete algorithm with a continuous proof:* our proof framework is not based on the discretization of an Ordinary Differential Equation (ODE) but rather studies *the evolution of a Stochastic Differential Equation (SDE) with jumps*. However, by contrast to (Even et al., 2021a), in our work, we significantly reduce the necessary amount of gradient steps compared to (Even et al., 2021a) while maintaining the same amount of activated edges. Another type of asynchronous algorithm can also be found in (Latz, 2021), yet it fails to obtain Nesterov's accelerated rates for lack of momentum. We note that (Leblond et al., 2018) studies the robustness to delays yet requires a shared memory and thus applies to a different context than decentralized optimization. (Hendrikx, 2022) is a promising approach for modeling random communication on graphs yet fails to obtain acceleration in a neat framework without inner loops.

Decentralized algorithms with fixed topology. (Scaman et al., 2017) is the first work to derive an accelerated algorithm for decentralized optimization, and it links the convergence speed to the Laplacian eigengap. The corresponding algorithm uses a dual formulation and a Chebychev acceleration (synchronous and only for fixed topology). Yet, as stated in Tab. 3.3, it still requires many edges to be activated. Furthermore, under a relatively flexible condition on the in-

[1] https://github.com/AdelNabli/DADAO/

Table 1. This table shows the strength of DADAO compared to concurrent works for obtaining ϵ-precision. n is the number of nodes, $|\mathcal{E}|$ the number of edges, $\frac{1}{\chi_1}$ the smallest positive eigenvalue of a fixed weighted Laplacian $\mathcal{L}$, ρ the eigengap and $\chi_2 \leq \chi_1$ the effective resistance. Note that under reasonable assumptions $\sqrt{\chi_1\chi_2}n = \mathcal{O}(|\mathcal{E}|\sqrt{\rho})$ (see Lemma 3.3). Async., Comm., Grad., M.-C. and Prox. stand respectively for Asynchrony, Communication steps, Gradient steps., Multi-consensus and Proximal operator. As suggested in their respective papers, all the algorithms are run with $\frac{1}{\|\mathcal{L}\|}\mathcal{L}$. For OGT, we note that the matrix is stochastic, a more precise comparison is given by Prop. 3.9.

Method	Async.	Decoupled	No Inner Loop (M.-C. or Prox.)	Primal Oracle	Total # Comm.	Total # Grad.
MSDA (Scaman et al., 2017)	✗	✗	✗	✗	$\sqrt{\rho}\|\mathcal{E}\|\sqrt{\frac{L}{\mu}}\log\frac{1}{\epsilon}$	$n\sqrt{\frac{L}{\mu}}\log\frac{1}{\epsilon}$
DVR (Hendrikx et al., 2020)	✗	✗	✗	✓	$\sqrt{\rho}\|\mathcal{E}\|\sqrt{\frac{L}{\mu}}\log\frac{1}{\epsilon}$	$n\sqrt{\frac{L}{\mu}}\log\frac{1}{\epsilon}$
ADOM+ (Kovalev et al., 2021a)	✗	✗	✗	✓	$\rho\|\mathcal{E}\|\sqrt{\frac{L}{\mu}}\log\frac{1}{\epsilon}$	$n\sqrt{\frac{L}{\mu}}\log\frac{1}{\epsilon}$
TVR (Hendrikx, 2022)	✗	✓	✗	✓	$\rho\|\mathcal{E}\|\frac{L}{\mu}\log\frac{1}{\epsilon}$	$n\frac{L}{\mu}\log\frac{1}{\epsilon}$
AGT (Li & Lin, 2021)	✗	✗	✗	✓	$\sqrt{\rho}\|\mathcal{E}\|\sqrt{\frac{L}{\mu}}\log\frac{1}{\epsilon}$	$n\sqrt{\frac{L}{\mu}}\log\frac{1}{\epsilon}$
OGT (Song et al., 2021)	✗	✓	✓	✓	$\sqrt{\rho}\|\mathcal{E}\|\sqrt{\frac{L}{\mu}}\log\frac{1}{\epsilon}$	$n\sqrt{\frac{L}{\mu}}\log\frac{1}{\epsilon}$
ESDACD (Hendrikx et al., 2019b)	✓	✗	✓	✗	$\sqrt{\chi_1\chi_2}n\sqrt{\frac{L}{\mu}}\log\frac{1}{\epsilon}$	$\sqrt{\chi_1\chi_2}n\sqrt{\frac{L}{\mu}}\log\frac{1}{\epsilon}$
Continuized (Even et al., 2021a)	✓	✗	✓	✗	$\sqrt{\chi_1\chi_2}n\sqrt{\frac{L}{\mu}}\log\frac{1}{\epsilon}$	$\sqrt{\chi_1\chi_2}n\sqrt{\frac{L}{\mu}}\log\frac{1}{\epsilon}$
DADAO (ours)	✓	✓	✓	✓	$\sqrt{\chi_1\chi_2}n\sqrt{\frac{L}{\mu}}\log\frac{1}{\epsilon}$	$n\sqrt{\frac{L}{\mu}}\log\frac{1}{\epsilon}$

tensity of our P.P.P.s, we show that our work improves over bounds that depend on the spectral gap. An emerging line of work following this formulation employs the continuized framework (Even et al., 2020; 2021a;b), but unfortunately do not use a primal oracle, as they rely on the gradients of the Fenchel conjugate. Finally, we note that the work of (Even et al., 2021b) incorporates delays in their model, showing that, with some adaptation, the continuized methods in (Even et al., 2021a) still converge at a linear rate. Yet transferring this robustness to our setting remains unclear. Reducing the number of communication has been studied in (Mishchenko et al., 2022), but without obtaining accelerated rates. (Hendrikx et al., 2021) allows for fast communication and gossip rates yet requires a proximal step and synchrony between nodes to apply a momentum variable.

Finite sum acceleration. If each local function f_i is a sum of elementary functions $\sum_{j=1}^m f_{i,j}$ with a favorable conditionning, an additional acceleration is possible, as observed by (Hendrikx et al., 2021; 2020; Hendrikx, 2022), which is a different problem from Eq. 1. This can be viewed as a cluster of nodes with infinite connectivity and an efficient decentralized algorithm should automatically adapt to such structure. Thus, we focused on the setting $m = 1$, which allows a fair comparison with these works.

Error feedback/Gradient tracking. A major lock for asynchrony is the use of Gradient Tracking (GT) (Koloskova et al., 2021; Nedic et al., 2017; Li & Lin, 2021) or Error Feedback (Stich & Karimireddy, 2020; Kovalev et al., 2021b). Indeed, gradient operations are locally tracked by a running-mean variable which must be synchronously updated at each gradient update (in GT, the sum of this distributed variable keeps track of the gradient of the objective function), making it incompatible with an asynchronous framework. (Zhang & You, 2019) uses a GT-*like* procedure, which do not verify this property, at the cost of using a buffer (increasing memory requirements) and worse convergence rates (*e.g.*, not accelerated). Furthermore, acceleration requires an undesirable multi-consensus inner loop. We emphasize that (Song et al., 2021) allows to decouple the gradient updates from communication, yet the framework is still synchronous, leading to synchronous communication rates, that asynchrony can improve (see Tab. 1).

Decoupling procedures. Decoupling subsequent steps of optimization procedures traditionally leads to speed-ups (Hendrikx et al., 2021; Hendrikx, 2022; Belilovsky et al., 2020; 2021). This contrasts with methods which couple gradient and gossip updates, so that they happen in a predefined order, i.e., simultaneously (Even et al., 2021a) or sequentially (Kovalev et al., 2021a; Koloskova et al., 2020). In decoupled optimization procedures, inner-loops are not desirable as they require an external procedure that can be potentially slow and need a block-barrier instruction during the algorithm's execution (e.g., (Hendrikx et al., 2021)). It means in particular that it is preferrable to avoid approaches such as Catalyst (Lin et al., 2015), multi-consensus steps (Kovalev et al., 2021c) or Tchebychev acceleration of consensus (Scaman et al., 2017).

Resistance of a graph. The maximal resistance of a graph is a widely studied quantity, particularly in physics (Klein & Randić, 1993; Vos, 2016; Klein, 2002), as it is a refined

geometric invariant of graphs. The resistance of a graph corresponds to the commute time of a Markov Chain (Chandra et al., 1996). However, beyond the Continuized framework (Even et al., 2021a) or acceleration of consensus problems (Can et al., 2022; Aybat & Gürbüzbalaban, 2017; Ghosh et al., 2008), we are unaware of generic, asynchronous, accelerated decentralized optimization procedures that rely on this quantity. Also, it has a more physical interpretation than the Laplacian's norm, as it can be computed via Ohm's and Kirchhoff's Circuit Laws (see (Chandra et al., 1996)).

3. Accelerated Asynchronous Algorithm

3.1. Gossip Framework

We consider the problem defined by Eq. 1 in a distributed environment constituted by n nodes whose dynamic is indexed by a continuous time index $t \in \mathbb{R}^+$. Each node has a local memory and can compute a local gradient ∇f_i, as well as elementary operations, in an instantaneous manner. As said above, having no delay is less realistic, yet adding them also leads to significantly more difficult proofs whose adaptation to our framework remains largely unclear. Next, we will assume that our computations and gossip result from independent homogeneous P.P.P. with no delay. For the sake of simplicity, we assume that all nodes can compute a gradient at the same rate:

Assumption 3.1 (Homogeneous gradient computations)**.** The gradient computations are normalized to fire independently at a rate of 1 computation per time unit. For the i-th worker, we write $N_i(t)$ the corresponding P.P.P. of rate 1, as well as $\mathbf{N}(t) = (N_i(t))_{i\leq n}$.

Remark 3.2. The P.P.P $N_i(t)$ on node i means that taking gradient steps at i are discrete events, but the time intervals between two events is a random variable following an exponential law of parameter 1. Thus, the expected waiting time between two gradient steps on the i-th worker is 1 time unit.

Next, we model the bandwidth of each connection. For an edge (i, j) belonging to $\in \mathcal{E}$, the set of edges of a graph we assume connected (3.5), we write $M_{ij}(t)$ the P.P.P. with rate $0 < \lambda_{ij} < \infty$. When this P.P.P. fires, both nodes share and update their local memories. The rate λ_{ij} is adjustable locally by machine i while λ_{ji} is controlled by machine j. While λ_{ij} and λ_{ji} may be different, we highlight that the communication process is symmetric, *i.e.* both nodes update their local memories when either one of the two corresponding P.P.Ps fires. Thus, in the corresponding undirected graph $\bar{\mathcal{E}}$, each edge (i, j) will fire at a rate of $\lambda_{ij} + \lambda_{ji}$. Given our notations, if $(i, j) \notin \mathcal{E}$, then the connection between (i, j) can be thought as a P.P.P. with intensity 0. Taking the λ_{ij} as edge weights, we introduce the subsequent graph Laplacian, which is the expected Laplacian of our graph:

$$\Lambda \triangleq \sum_{(i,j)\in\mathcal{E}} \lambda_{ij}(e_i - e_j)(e_i - e_j)^\mathsf{T} .$$

We write $\mathbf{\Lambda} \triangleq \sum_{(i,j)\in\mathcal{E}} \lambda_{ij}(\mathbf{e}_i - \mathbf{e}_j)(\mathbf{e}_i - \mathbf{e}_j)^\mathsf{T}$ its tensorized counter-part that will be useful for our Lyapunov-based proofs. Following (Scaman et al., 2017), we will compare this quantity to the following projector:

$$\pi \triangleq \mathbf{I} - \frac{1}{n}\mathbf{1}\mathbf{1}^\mathsf{T} = \frac{1}{2n}\sum_{1\leq i,j\leq n}(\mathbf{e}_i - \mathbf{e}_j)(\mathbf{e}_i - \mathbf{e}_j)^\mathsf{T} .$$

In this context, a natural quantity is the algebraic connectivity of our network given by (Kovalev et al., 2021a):

$$\chi_1 \triangleq \sup_{\|x\|=1, x\perp\mathbf{1}} \frac{1}{x^\mathsf{T}\Lambda x} .$$

We might also write $\chi_1[\Lambda]$ to avoid confusion, depending on the context.

Next, the maximal effective resistance of the network, as in (Even et al., 2021a; Ellens et al., 2011), is:

$$\chi_2 \triangleq \frac{1}{2}\sup_{(i,j)\in\mathcal{E}} (e_i - e_j)^\mathsf{T}\Lambda^+(e_i - e_j) .$$

A standard quantity (Scaman et al., 2017), which is used to control the number of synchronous gossips steps, is the spectral gap, given by:

$$\rho \triangleq \|\Lambda\|\chi_1 .$$

We also introduce the value κ, the ratio of communication frequency between the fastest and slowest edges:

$$\kappa \triangleq \frac{\sup_{(i,j)\in\mathcal{E}} \lambda_{ij} + \lambda_{ji}}{\inf_{(i,j)\in\mathcal{E}} \lambda_{ij} + \lambda_{ji}} .$$

This ratio is typically bounded, for instance, in the case of a graph with constant edge weights or for $\lambda_{ij} = \frac{1}{d_i}$ with d_i the degree of the i-th node in a bounded degree graph or a regular graph.

We prove the following Lemma (proved in Appendix A), which is useful to control χ_1, χ_2 and compare our bounds with works that rely on the spectral gap of a graph:

Lemma 3.3 (Effective resistance)**.** *The spectrum of Λ is non-negative. Also, we have $\chi_1 = +\infty$ iff $\bar{\mathcal{E}}$ is not a connected graph. If the graph is connected, then:*

$$\frac{n-1}{\mathrm{Tr}\,\Lambda} \leq \chi_2 \leq \chi_1 .$$

Furthermore, we also have the following:

$$\sqrt{\chi_1\chi_2}\mathrm{Tr}\,\Lambda \leq \sqrt{\rho}\sqrt{\kappa n|\bar{\mathcal{E}}|} .$$

The last part of this Lemma indicates that our method requires less communications than methods depending on the spectral gap when no degenerated behavior on the graph's connectivity happens, *i.e.* when κ is adequately bounded, which is a standard assumption (Hendrikx et al., 2019a).

Remark 3.4. For synchronous frameworks (Kovalev et al., 2021a; Scaman et al., 2017), the spectral quantity extracted from their gossip matrix is a given measure of the connectedness of their graphs that is used afterwards to deduce the right number of synchronous communication rounds between two gradient steps. In our framework, Λ directly contains the information of both the topology $\mathcal{E}$ and the edge communication rates λ_{ij}, thus $\chi_1[\Lambda]$ must rather be understood as an indicator of how well the graphs connectedness and the chosen communication strategy interact. In fact, we will see later that there is a condition on χ_1, χ_2 for DADAO to converge, see Remark 3.7 for further discussion.

Assumption 3.5 (Connected graph)**.** The set of edges $\mathcal{E}$ defines a connected graph such that $\chi_1[\Lambda] < \infty$.

3.2. Dynamic to optimum

Next, we follow a standard approach (Kovalev et al., 2020; 2021a; Salim et al., 2022; Hendrikx, 2022) for solving Eq. 1 (see Appendix B for details), leading to studying, for $0 < \nu < \mu$, the following Lagrangian:

$$\inf_{x\in\mathbb{R}^{n\times d}} \sup_{\substack{y\in\mathbb{R}^{n\times d}\\ z\in\mathbb{R}^{n\times d}}} \sum_{i=1}^{n} f_i(x_i) - \frac{\nu}{2}\|x\|^2 - \langle x, y\rangle - \frac{1}{2\nu}\|\pi z + y\|^2.$$

For $f(x) = \sum_{i=1}^n f_i(x_i)$, the saddle points (x^*, y^*, z^*) of the above Lagrangian are given by:

$$\begin{cases} \nabla f(x^*) - \nu x^* - y^* &= 0 \\ y^* + \pi z^* + \nu x^* &= 0 \\ \pi z^* + \pi y^* &= 0\,. \end{cases} \tag{2}$$

Our algorithm is based on a fixed-point algorithm to obtain those saddle points, which is a similar idea to (Kovalev et al., 2021b), which is only restricted to a setting without communication acceleration. Furthermore, contrary to (Kovalev et al., 2021a), we do not employ a Forward-Backward algorithm, which requires both an extra-inversion step and additional regularity on the considered proximal operator. Not only does this condition not hold in this particular case, but this is not desirable in a continuized framework where iterates are not ordered in a predefined sequence and require a local descent at each instant. Another major difference is that no Error-Feedback is required by our approach, which allows unlocking asynchrony while simplifying the proofs and decreasing the required number of communications. Instead, we show it is enough to incorporate a standard fixed point algorithm, *without any specific preconditioning* (see (Condat et al., 2019)). We consider the following dynamic:

$$\begin{aligned}
dx_t &= \eta(\tilde{x}_t - x_t)dt - \gamma(\nabla f(x_t) - \nu x_t - \tilde{y}_t)\, d\mathbf{N}(t)\\
d\tilde{x}_t &= \tilde{\eta}(x_t - \tilde{x}_t)dt - \tilde{\gamma}(\nabla f(x_t) - \nu x_t - \tilde{y}_t)\, d\mathbf{N}(t)\\
d\tilde{y}_t &= -\theta(y_t + z_t + \nu\tilde{x}_t)dt\\
&\quad + (\delta + \tilde{\delta})(\nabla f(x_t) - \nu x_t - \tilde{y}_t)d\mathbf{N}(t)\\
dy_t &= \alpha(\tilde{y}_t - y_t)dt\\
dz_t &= \alpha(\tilde{z}_t - z_t)dt\\
&\quad - \beta \sum_{(i,j)\in\mathcal{E}} (\mathbf{e}_i - \mathbf{e}_j)(\mathbf{e}_i - \mathbf{e}_j)^\mathsf{T}(y_t + z_t)dM_{ij}(t)\\
d\tilde{z}_t &= \tilde{\alpha}(z_t - \tilde{z}_t)dt\\
&\quad - \tilde{\beta} \sum_{(i,j)\in\mathcal{E}} (\mathbf{e}_i - \mathbf{e}_j)(\mathbf{e}_i - \mathbf{e}_j)^\mathsf{T}(y_t + z_t)dM_{ij}(t)\,,
\end{aligned} \tag{3}$$

where $\nu, \tilde{\eta}, \eta, \gamma, \alpha, \tilde{\alpha}, \theta, \delta, \tilde{\delta}, \beta, \tilde{\beta}$ are undetermined real-valued parameters. As in (Nesterov, 2003), variables are paired to obtain a Nesterov acceleration. The variables (x, y) allow decoupling the gossip steps from the gradient steps using independent P.P.P.s. Furthermore, the Lebesgue integrable path of $\tilde{y}_t$ does not correspond to a standard momentum, as in a continuized framework (Even et al., 2021a); however, it turns out to be a crucial component of our method. Compared to (Kovalev et al., 2021a), no extra multi-consensus step needs to be integrated. Our formulation of an asynchronous gossip step is similar to (Even et al., 2021a), which introduces a stochastic variable on edges; however, contrary to this work, our gossip and gradient computations are decoupled. We emphasize that while the dynamic 3 is formulated using SDEs (Arnold, 1974), which brings the power of the continuous-time analysis toolbox, it is still *event-based* and thus discrete in nature. Hence, the dynamic can be efficiently implemented in practice as explained in Sec. 4.1 and Appendix E.

3.3. Theoretical guarantees

We prove the following in Appendix C.

Theorem 3.6. *Assume each f_i is μ-strongly convex and L-smooth. Assume 3.1, 3.5, and that $2\chi_1\chi_2 \leq 1$. Then there exists some parameters for the dynamic Eq.* (3) *(given in Appendix C.1), such that for any initialization $x_0 \in ker(\pi)$, and $\tilde{x}_0 = x_0, y_0 = \tilde{y}_0 = \nabla f(x_0) - \frac{\mu}{2}x_0, z_0 = \tilde{z}_0 = -\pi y_0$, we get for $t \in \mathbb{R}^+$:*

$$\mathbb{E}[\|x_t - x^*\|^2] \leq (\frac{1}{2} + \frac{23}{8}\frac{L}{\mu} + 2\frac{L^2}{\mu^2})\|x_0 - x^*\|^2 e^{-\frac{t}{8\sqrt{2}}\sqrt{\frac{\mu}{L}}}$$

Also, the expected number of oracle gradient calls is nt and the expected number of edges activated is: $\frac{t}{2}\mathrm{Tr}\,\Lambda$.

The condition $2\chi_1\chi_2 \leq 1$ can simply be understood as whether or not the chosen communication strategy suits sufficiently well the graph topology $\mathcal{E}$ for DADAO to converge

using the expected rates of communication $(\lambda_{ij})_{(i,j)\in\mathcal{E}}$ compared to the expected rate of one gradient step per time unit on each node we assumed in Assumption 3.1. In fact, we make the following interpretation:

Remark 3.7. Introducing $\lambda \triangleq \sum_{(ij)\in\mathcal{E}} \lambda_{ij}$ and $\mathcal{P}_\Lambda \triangleq \frac{2\Lambda}{\mathrm{Tr}\ \Lambda}$, we write Λ as the product of these two more interpretable quantities to gain more insight on the condition $2\chi_1[\Lambda]\chi_2[\Lambda] \leq 1$:

$$\Lambda = \lambda \mathcal{P}_\Lambda. \tag{4}$$

In this setting, λ is the expected rate of communication over the *whole graph*, while $\mathcal{P}_\Lambda$ can be interpreted as the Laplacian of $\mathcal{E}$ with each edge weighted with its probability of having spiked given a communication happened in the graph. We have:

$$\chi_1[\Lambda] = \frac{\chi_1[\mathcal{P}_\Lambda]}{\lambda} \quad ; \quad \chi_2[\Lambda] = \frac{\chi_2[\mathcal{P}_\Lambda]}{\lambda}. \tag{5}$$

$\mathcal{P}_\Lambda$ being normalized, we could say that the quantities $\chi_1[\mathcal{P}_\Lambda], \chi_2[\mathcal{P}_\Lambda]$ characterize the graph's connectivity while λ is the global rate of communication. Then, using (5), the condition $2\chi_1[\Lambda]\chi_2[\Lambda] \leq 1$ is equivalent to saying

$$\sqrt{2\chi_1[\mathcal{P}_\Lambda]\chi_2[\mathcal{P}_\Lambda]} \leq \lambda, \tag{6}$$

meaning that the global communication rate should be larger than some spectral quantity quantifying the graph's connectivity. If structural constraints on the network makes it impossible to verify this condition, as the notion of time is *only* defined through the rate of gradient steps given by Assumption 3.1 (λ can be interpreted as *"the expected number of communications in the graph between the expected duration separating two subsequent gradient steps on a given node"*), it only means that the gradient steps are happening too fast compared to the ability of the network to communicate, and the rate of gradient steps should be slow-down by a factor of $\sqrt{2\chi_1\chi_2}$.

Thus, given a graph topology, it is straightforward to obtain a Laplacian verifying $2\chi_1\chi_2 \leq 1$. Indeed, we have the following:

Remark 3.8. For any graph topology $\mathcal{E}$ and any choice of corresponding graph Laplacian $\mathcal{L}$ verifying $\chi_1[\mathcal{L}]$ bounded, there is a communication strategy $(\lambda_{ij})_{(i,j)\in\mathcal{E}}$ such that Λ, the Laplacian of $\mathcal{E}$ with edges $(i,j) \in \mathcal{E}$ weighted with their expected communication rate λ_{ij}, verifies $2\chi_1[\Lambda]\chi_2[\Lambda] \leq 1$. One such example is given by:

$$\Lambda = \sqrt{2\chi_1[\mathcal{L}]\chi_2[\mathcal{L}]}\mathcal{L}\,.$$

This property allows us to use in DADAO the Laplacians introduced in previous work, to which we can now compare. It leads to the following proposition (proved in Appendix D), which shows that DADAO obtains better complexities than concurrent works while starting from the same family of Laplacians:

Proposition 3.9 (Comparison with concurrent work)**.**

- *If $\kappa = \mathcal{O}(\frac{|\bar{\mathcal{E}}|}{n})$, then DADAO obtains better communication rate than MSDA (Scaman et al., 2017), and requires strictly less communications for the complete graph.*
- *For any fixed Laplacian valid for ADOM+ (Kovalev et al., 2021a), DADAO obtains a better communication rate than ADOM+, and requires strictly less communications for the complete graph.*
- *If $\kappa = \mathcal{O}(\frac{|\bar{\mathcal{E}}|}{n})$, for a valid Laplacian using the Gossip matrix of OGT (Song et al., 2021) or AGT (Li & Lin, 2021), DADAO obtains a better communication rate than both Gradient Tracking methods, and requires strictly less communications for the complete graph.*
- *DADAO requires fewer gradient computations than the Continuized framework (Even et al., 2021b), and has a strictly better computation rate for the cycle graph.*

We highlight that (Scaman et al., 2017) claimed that their algorithm is optimal because they study the number of computations and *synchronized* gossips on a worst-case graph; our claim is, *by nature* different, as we are interested in the number of edges fired rather than the number of synchronized gossip rounds. Indeed, in an asynchronous framework, there is no notion of *round of* communication, which allows this framework to enjoy the advantageous rates of randomized procedures. Moreover, not only this measure of complexity is standard in asynchronous frameworks (*e.g.*, see (Boyd et al., 2006)), but also, if we are aiming for the most frugal procedure, minimizing both the total number of computations and communications is of interest. Tab. 3.3 predicts the behavior of our algorithm on various classes of graphs encoded via a normalized Laplacian. It shows that systematically, our algorithm leads to the best complexities. For example, in the case of a complete graph, one synchronized gossip round requires a total of $|\mathcal{E}| = \mathcal{O}(n^2)$ communications whereas we show that a rate of only $\mathcal{O}(n)$ communications per time unit suffices for DADAO in this case. We note that the graph class depicted in Tab. 3.3 was used as worst-case examples for proving the optimality of (Scaman et al., 2017) in a synchronous context.

4. Practical implementation

4.1. Algorithm

To study the trajectories of $X_t \triangleq (x_t, \tilde{x}_t, \tilde{y}_t), Y_t \triangleq (y_t, z_t, \tilde{z}_t)$, we use the following, equivalent to (3):

$$\begin{aligned} dX_t &= a_1(X_t, Y_t)dt + b_1(X_t)d\mathbf{N}(t) \\ dY_t &= a_2(X_t, Y_t)dt + \sum_{(i,j)\in\mathcal{E}} b_2^{ij}(Y_t)dM_{ij}(t)\,, \end{aligned} \tag{7}$$

Table 2. Complexity for various graphs using $\frac{1}{\|\mathcal{L}\|}\mathcal{L}$ with $\mathcal{L}$ the standard Laplacian with unit edge weights. In this case, $\rho = \chi_1$. The complexities are reported per time unit so that all algorithms reach ϵ-precision at the same time. We have, respectively, for a star/line or cyclic/complete graph and the d-dimensional grid: $\chi_1 = \mathcal{O}(n)$, $\chi_2 = \mathcal{O}(n)$, $\mathrm{Tr} = \mathcal{O}(1)$ / $\chi_1 = \mathcal{O}(n^2)$, $\chi_2 = \mathcal{O}(1)$, $\mathrm{Tr} = \mathcal{O}(n)$ / $\chi_1 = \mathcal{O}(1)$, $\chi_2 = \mathcal{O}(1)$, $\mathrm{Tr} = \mathcal{O}(n)$ / $\chi_1 = \mathcal{O}(n^{2/d})$, $\chi_2 = \mathcal{O}(1)$, $\mathrm{Tr} = \mathcal{O}(n)$.

Method	# edges activated per time unit				# gradients computed per time unit			
Graph	Star	Line	Complete	d-grid	Star	Line	Complete	d-grid
(Kovalev et al., 2021a) ADOM+	n^2	n^3	n^2	$n^{1+2/d}$	n	n	n	n
(Scaman et al., 2017) MSDA	$n^{3/2}$	n^2	n^2	$n^{1+1/d}$	n	n	n	n
(Even et al., 2021a) Continuized	n	n^2	n	$n^{1+1/d}$	n	n^2	n	$n^{1+1/d}$
Centralized	n	-	-	-	n	-	-	-
DADAO (ours)	n	n^2	n	$n^{1+1/d}$	n	n	n	n

where $a_1, a_2, b_1 = (b_1^i)_i, (b_2^{ij})_{ij}$ are smooth functions. We now describe the algorithm used to implement the dynamics of Eq. (3). Let us write $T_1^{(i)} < T_2^{(i)} < ... < T_k^{(i)} < ...$ the time of the k-th event on the i-th node, which is either an edge activation, either a gradient update. We remind that the spiking times of a specific event correspond to random variables with independent exponential increments and can thus be generated at the beginning of our simulation. They can also be generated on the fly and locally to stress the locality and asynchronicity of our algorithm. We write $X_t = (X_t^{(i)})_i$ and $Y_t = (Y_t^{(i)})_i$, then on the i-th node, at the k-th iteration, we integrate on $[T_k^{(i)}; T_{k+1}^{(i)}]$ the ODE

$$dX_t = a_1(X_t, Y_t)dt$$
$$dY_t = a_2(X_t, Y_t)dt$$

to define the values right before the spike. One can easily find the matrix $\mathcal{A} \in \mathbb{R}^{6\times 6}$ such that:

$$\begin{pmatrix} X^{(i)}_{T^{(i)-}_{k+1}} \\ Y^{(i)}_{T^{(i)-}_{k+1}} \end{pmatrix} = \exp\left((T^{(i)}_{k+1} - T^{(i)}_k)\mathcal{A}\right) \begin{pmatrix} X^{(i)}_{T^{(i)}_k} \\ Y^{(i)}_{T^{(i)}_k} \end{pmatrix} . \quad (8)$$

Next, if one has a gradient update, then:

$$X^{(i)}_{T^{(i)}_{k+1}} = X^{(i)}_{T^{(i)-}_{k+1}} + b_1\left(X^{(i)}_{T^{(i)-}_{k+1}}\right) .$$

Otherwise, if the edge (i, j) or (j, i) is activated, a communication bridge is created between both nodes i and j. In this case, the local update on i writes:

$$Y^{(i)}_{T^{(i)}_{k+1}} = Y^{(i)}_{T^{(i)-}_{k+1}} + b_2\left(Y^{(i)}_{T^{(i)-}_{k+1}}, Y^{(j)}_{T^{(i)-}_{k+1}}\right) .$$

Note that, even if this event takes place along an edge (i, j), we can write it separately for nodes i and j by making sure they both have the events $T^{(i)}_{k_i} = T^{(j)}_{k_j}$, for some $k_i, k_j \in \mathbb{N}$ corresponding to this communication. As advocated, all those operations are local, and we summarize in the Alg. 1 the algorithmic block corresponding to our implementation. See Appendix E for more details.

Algorithm 1: This algorithm block describes our implementation on each local machine. The ODE routine is described by Eq. 8.

Input: On each machine $i \in \{1, ..., n\}$, gradient oracle ∇f_i, parameters $\mu, L, \chi_1, t_{\max}$.
Initialize on each machine $i \in \{1, ..., n\}$:
  Set $X^{(i)}, Y^{(i)}, T^{(i)}$ to 0 and set $\mathcal{A}$ via Eq. (75);
Synchronize the clocks of all machines ;
In parallel *on workers* $i \in \{1, ..., n\}$, **while** $t < t_{\max}$, **continuously do:**
  $t \leftarrow clock()$;
  if *there is an event at time* t **then**
    $(X^{(i)}, Y^{(i)}) \leftarrow ODE(\mathcal{A}, t - T^{(i)}, X^{(i)}, Y^{(i)})$;
    if *the event is to take a gradient step* **then**
      $X^{(i)} \leftarrow X^{(i)} + b_1(X^{(i)})$;
    else if *the event is to communicate with* j **then**
      $Y^{(i)} \leftarrow Y^{(i)} + b_2(Y^{(i)}, Y^{(j)})$;
      `// Happens at j simultaneously.`
    $T^{(i)} \leftarrow t$;
return $(x^{(i)}_{t_{\max}})_{1\le i\le n}$.

4.2. Numerical results

In this section, we study the behavior of our method in a standard experimental setting (*e.g.*, see(Kovalev et al., 2021a; Even et al., 2021a)). In order to compare to methods using the gradient of the Fenchel conjugate (Even et al., 2021a) in our experiments, we restrict ourselves to a situation where it is easily computable. Thus, we perform the empirical risk minimization for the decentralized linear regression task given by:

$$f_i(x) = \frac{1}{m}\sum_{j=1}^{m} \|a_{ij}^\top x - c_{ij}\|^2, \quad (9)$$

where $a_{ij} \in \mathbb{R}^d$, and $c_{ij} \in \mathbb{R}$ correspond to m local data points stored at node i. We follow a protocol similar to (Kovalev et al., 2021a): we generate n independent synthetic datasets with the `make_regression` functions of scikit-

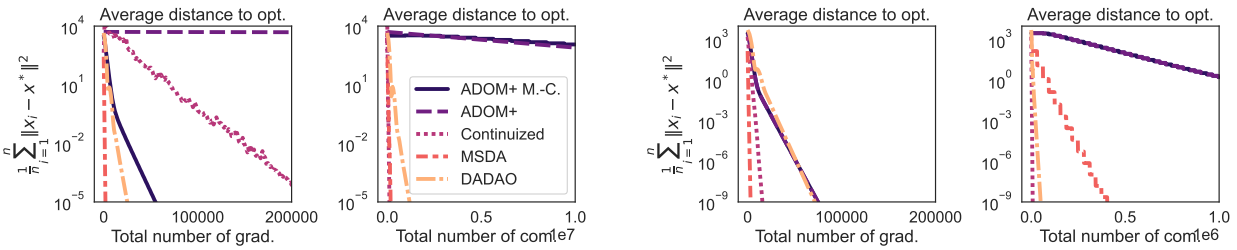

Figure 1. Comparison between ADOM+ (Kovalev et al., 2021a), the continuized framework (Even et al., 2021a), MSDA (Scaman et al., 2017) and DADAO, using the same data for the linear regression task, and the same graphs *(from left to right: line with $n = 150$, complete with $n = 250$).*

learn (Pedregosa et al., 2011), each worker storing $m = 100$ data points. We recall that the metrics of interest are the total number of local gradient steps and the total number of individual messages exchanged *(*i.e.*, number of edges that fired)* to reach an ϵ-precision. We systematically used the proposed hyper-parameters of each reference paper for our implementation without any specific fine-tuning.

Comparison between all methods. We fix the Laplacian matrix via Eq. (3.8) to compare simultaneously to the continuized framework (Even et al., 2021a), MSDA (Scaman et al., 2017) and ADOM+ (Kovalev et al., 2021a). We compare ourselves to both versions of ADOM+: with and without the Multi-Consensus (M.-C.). We report in Fig. 1 results corresponding to the complete graph with $n = 250$ nodes and the line graph of size $n = 150$. While sharing the same asymptotic rate (see Fig. 2 for experimental confirmation), we note that the Continuized framework (Even et al., 2021a) and MSDA (Scaman et al., 2017) have better absolute constants than DADAO, giving them an advantage both in terms of the number of communication and gradient steps. However, in the continuized framework, the gradient and communication steps being coupled, the number of gradient computations can potentially be orders of magnitude worse than our algorithm, which is reflected by Fig. 1 for the line graph. As for MSDA, Tab. 3.3 showed they do not have the best communication rates on certain classes of graphs, as confirmed to the right in Fig. 1 for MSDA and in Fig. 2. Thanks to its M.-C. procedure, ADOM+ can significantly reduce the number of necessary gradient steps. Yet, consistently with our analysis in Prop. 3.9, our method is systematically better in all settings in terms of communications.

Further comparison between DADAO and MSDA. To experimentally confirm that our communication complexity is better than accelerated methods using the spectral gap and abstract away the better absolute constants for MSDA, we ran DADAO and MSDA on the task of distributed linear regression for star graphs of size $n \in \{10, 20, 70, 200, 300, 1000, 2000\}$. We considered the evolution of the average distance to the optimal with the number of gradient steps and commmunication steps in log scale for each run, and computed the slope of each line. For each graph size, we report in Fig. 2 the rate between the slope for DADAO and the slope for MSDA. We remark that the rate

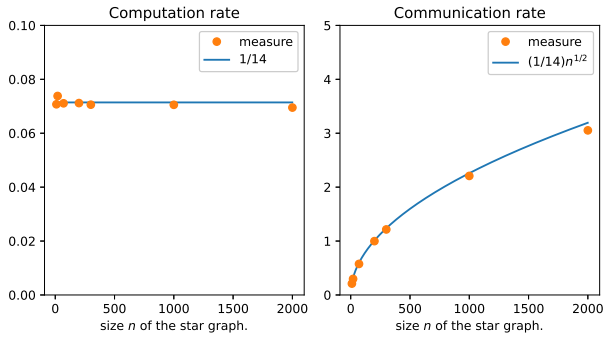

Figure 2. Rate between the slopes in log scale of DADAO and MSDA for star graphs of size $n \in \{10, 20, 70, 200, 300, 1000, 2000\}$.

between the gradient complexities of DADAO and MSDA is indeed a $\mathcal{O}(1)$ (with a constant value of $\simeq 1/14$) while MSDA is indeed $\mathcal{O}(\sqrt{n})$ worse than DADAO for communications on the star graph, as stated in Tab. 3.3.

Discussion. We do not claim the optimality of the absolute constant in DADAO's rate (the $8\sqrt{2}$ term in the exponential of Th. 3.6), which might be further optimized with a tighter analysis and different values of parameters. However, our focus being on large scale problems, we are concerned about asymptotic rates rather than the effect of the absolute constants. Thus, even-though Fig. 1 may sometimes display a faster convergence for the continuized framework (Even et al., 2021a) and MSDA (Scaman et al., 2017) in terms of *number of steps*, we remind that both (Even et al.,

2021a; Scaman et al., 2017) use expensive evaluation of dual gradients whereas we only use primal gradients, and Tab. 3.3 confirms that DADAO has at least their asymptotic rate. On the star graph, where Tab. 3.3 showed a strict advantage for DADAO compared to MSDA, Fig. 2 confirms that our convergence rate meets the one of (Scaman et al., 2017) for $n \simeq 200$, and has strictly better rates for larger graphs. Finally, we note that, our algorithm falling into the *randomized* category, further improvements may be theoretically possible. Indeed, (Woodworth & Srebro, 2016) showed that the lower bound on the number of grad-and-prox oracle accesses for the optimization of composite objectives with deterministic methods is in $\mathcal{O}(n\sqrt{\frac{L}{\mu}}\log(\frac{1}{\epsilon}))$, which is attained by work such as (Scaman et al., 2017). However, for randomized algorithms, the lower bound is in $\mathcal{O}(n+\sqrt{\frac{nL}{\mu}}\log(\frac{1}{\epsilon}))$ and, to the best of our knowledge, no algorithm meeting this bound is known to date in the decentralized setting, and in particular, in the asynchronous one.

In conclusion, while several methods can share similar asymptotic convergence rates, ours is the only one to perform at least as well as its competitors in every setting for different graph's topology, as predicted by Tab. 1.

5. Conclusion

In this work, we have proposed a novel stochastic algorithm for the decentralized optimization of a sum of smooth and strongly convex functions. We have demonstrated, theoretically and empirically, that this algorithm leads systematically to a substantial acceleration compared to state-of-the-art works. Furthermore, our algorithm is asynchronous, decoupled, primal, and does not rely on an extra inner loop: each of these properties makes it suitable for real applications. In future work, we would like to explore the robustness of such algorithms to more challenging variabilities occurring in real-life applications such as time-varying networks and to extend our work to less regular functions and stochastic analysis.

Acknowledgements

This work was supported by Project ANR-21-CE23-0030 ADONIS and EMERG-ADONIS from Alliance SU. This work was granted access to the HPC/AI resources of IDRIS under the allocation AD011013743 made by GENCI. In addition, the authors would like to thank Raphaël Berthier, Mathieu Even, Hadrien Hendrikx, and Dmitry Kovalev for their helpful discussions.

References

Arnold, L. Stochastic differential equations. *New York*, 1974.

Aybat, N. S. and Gürbüzbalaban, M. Decentralized computation of effective resistances and acceleration of consensus algorithms. In *2017 IEEE Global Conference on Signal and Information Processing (GlobalSIP)*, pp. 538–542, 2017. doi: 10.1109.

Belilovsky, E., Eickenberg, M., and Oyallon, E. Decoupled greedy learning of CNNs. In III, H. D. and Singh, A. (eds.), *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 736–745. PMLR, 13–18 Jul 2020.

Belilovsky, E., Leconte, L., Caccia, L., Eickenberg, M., and Oyallon, E. Decoupled greedy learning of cnns for synchronous and asynchronous distributed learning. *arXiv preprint arXiv:2106.06401*, 2021.

Boyd, S., Ghosh, A., Prabhakar, B., and Shah, D. Randomized gossip algorithms. *IEEE Transactions on Information Theory*, 52(6):2508–2530, 2006. doi: 10.1109/TIT.2006.874516.

Can, B., Soori, S., Aybat, N. S., Dehnavi, M. M., and Gürbüzbalaban, M. Randomized gossiping with effective resistance weights: Performance guarantees and applications. *IEEE Transactions on Control of Network Systems*, 9(2):524–536, 2022.

Chandra, A. K., Raghavan, P., Ruzzo, W. L., Smolensky, R., and Tiwari, P. The electrical resistance of a graph captures its commute and cover times. *computational complexity*, 6(4):312–340, 1996.

Condat, L., Kitahara, D., Contreras, A., and Hirabayashi, A. Proximal splitting algorithms for convex optimization: A tour of recent advances, with new twists, 2019.

Ellens, W., Spieksma, F. M., Van Mieghem, P., Jamakovic, A., and Kooij, R. E. Effective graph resistance. *Linear algebra and its applications*, 435(10):2491–2506, 2011.

Even, M., Hendrikx, H., and Massoulié, L. Asynchrony and acceleration in gossip algorithms, 2020.

Even, M., Berthier, R., Bach, F., Flammarion, N., Hendrikx, H., Gaillard, P., Massoulié, L., and Taylor, A. A continuized view on nesterov acceleration for stochastic gradient descent and randomized gossip. In Beygelzimer, A., Dauphin, Y., Liang, P., and Vaughan, J. W. (eds.), *Advances in Neural Information Processing Systems*, 2021a.

Even, M., Hendrikx, H., and Massoulie, L. Decentralized optimization with heterogeneous delays: a continuous-time approach. *arXiv preprint arXiv:2106.03585*, 2021b.

Ghosh, A., Boyd, S., and Saberi, A. Minimizing effective resistance of a graph. *SIAM Review*, 50(1):37–66, 2008. doi: 10.1137/050645452.

Hendrikx, H. A principled framework for the design and analysis of token algorithms, 2022.

Hendrikx, H., Bach, F., and Massoulié, L. An accelerated decentralized stochastic proximal algorithm for finite sums. In Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019a.

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

Hendrikx, H., Bach, F., and Massoulié, L. Dual-free stochastic decentralized optimization with variance reduction. *Advances in neural information processing systems*, 33: 19455–19466, 2020.

Hendrikx, H., Bach, F., and Massoulié, L. An optimal algorithm for decentralized finite-sum optimization. *SIAM Journal on Optimization*, 31(4):2753–2783, 2021. doi: 10.1137/20M134842X.

Klein, D. J. Resistance-distance sum rules. *Croatica chemica acta*, 75(2):633–649, 2002.

Klein, D. J. and Randić, M. Resistance distance. *Journal of mathematical chemistry*, 12(1):81–95, 1993.

Koloskova, A., Loizou, N., Boreiri, S., Jaggi, M., and Stich, S. A unified theory of decentralized sgd with changing topology and local updates. In *International Conference on Machine Learning*, pp. 5381–5393. PMLR, 2020.

Koloskova, A., Lin, T., and Stich, S. U. An improved analysis of gradient tracking for decentralized machine learning. *Advances in Neural Information Processing Systems*, 34:11422–11435, 2021.

Kovalev, D., Salim, A., and Richtarik, P. Optimal and practical algorithms for smooth and strongly convex decentralized optimization. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., and Lin, H. (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 18342–18352. Curran Associates, Inc., 2020.

Kovalev, D., Gasanov, E., Gasnikov, A., and Richtárik, P. Lower bounds and optimal algorithms for smooth and strongly convex decentralized optimization over time-varying networks. In Beygelzimer, A., Dauphin, Y., Liang, P., and Vaughan, J. W. (eds.), *Advances in Neural Information Processing Systems*, 2021a.

Kovalev, D., Gasnikov, A., and Richtárik, P. Accelerated primal-dual gradient method for smooth and convex-concave saddle-point problems with bilinear coupling. *arXiv preprint arXiv:2112.15199*, 2021b.

Kovalev, D., Shulgin, E., Richtárik, P., Rogozin, A. V., and Gasnikov, A. Adom: accelerated decentralized optimization method for time-varying networks. In *International Conference on Machine Learning*, pp. 5784–5793. PMLR, 2021c.

Last, G. and Penrose, M. *Lectures on the Poisson process*, volume 7. Cambridge University Press, 2017.

Latz, J. Analysis of stochastic gradient descent in continuous time. *Statistics and Computing*, 31(4):1–25, 2021.

Leblond, R., Pedregosa, F., and Lacoste-Julien, S. Improved asynchronous parallel optimization analysis for stochastic incremental methods. *arXiv preprint arXiv:1801.03749*, 2018.

Li, H. and Lin, Z. Accelerated gradient tracking over time-varying graphs for decentralized optimization. *arXiv preprint arXiv:2104.02596*, 2021.

Lin, H., Mairal, J., and Harchaoui, Z. A universal catalyst for first-order optimization. *Advances in neural information processing systems*, 28, 2015.

Mishchenko, K., Malinovsky, G., Stich, S., and Richtárik, P. Proxskip: Yes! local gradient steps provably lead to communication acceleration! finally! *arXiv preprint arXiv:2202.09357*, 2022.

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

Nesterov, Y. *Introductory lectures on convex optimization: A basic course*, volume 87. Springer Science & Business Media, 2003.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., and Chintala, S. Pytorch: An imperative style, high-performance deep learning library. In Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché-Buc,

F., Fox, E., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 32*, pp. 8024–8035. Curran Associates, Inc., 2019.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

Salim, A., Condat, L., Kovalev, D., and Richtárik, P. An optimal algorithm for strongly convex minimization under affine constraints. In *International Conference on Artificial Intelligence and Statistics*, pp. 4482–4498. PMLR, 2022.

Scaman, K., Bach, F., Bubeck, S., Lee, Y. T., and Massoulié, L. Optimal algorithms for smooth and strongly convex distributed optimization in networks. In Precup, D. and Teh, Y. W. (eds.), *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pp. 3027–3036. PMLR, 06–11 Aug 2017.

Song, Z., Shi, L., Pu, S., and Yan, M. Optimal gradient tracking for decentralized optimization. *arXiv preprint arXiv:2110.05282*, 2021.

Stich, S. U. and Karimireddy, S. P. The error-feedback framework: Sgd with delayed gradients. *Journal of Machine Learning Research*, 21(237):1–36, 2020.

Vos, V. S. S. Methods for determining the effective resistance. *Mestrado, Mathematisch Instituut Universiteit Leiden*, 2016.

Woodworth, B. E. and Srebro, N. Tight complexity bounds for optimizing composite objectives. In Lee, D., Sugiyama, M., Luxburg, U., Guyon, I., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc., 2016.

Zhang, J. and You, K. Fully asynchronous distributed optimization with linear convergence in directed networks, 2019.

Appendix

Table of Contents

A Communication bounds **12**

B Saddle Point Reformulation **13**

C Proof of the main Theorem (Th. 3.5) **13**

C.1 Proof of the Lemma C.1 15

D Comparison of complexities with related work **19**

E Practical Implementation **22**

E.1 Simulating the Poisson Point Processes 22

E.2 Pseudo Code 22

A. Communication bounds

The following properties will be used all along the proofs of the Lemmas and Theorems and are related to the communication of our nodes.

Lemma A.1. *Under the assumptions of Theorem 3.6, if $z_0, \tilde{z}_0 \in$ span(π), then $z_t, \tilde{z}_t \in$ span(π) almost surely.*

Proof. It's clear that for any i, j, we get:

$$\pi(\mathbf{e}_i - \mathbf{e}_j)(\mathbf{e}_i - \mathbf{e}_j)^\mathsf{T} = (\mathbf{e}_i - \mathbf{e}_j)(\mathbf{e}_i - \mathbf{e}_j)^\mathsf{T}.$$

Thus, the variations of $(z_t, \tilde{z}_t)$ belong to span(π), and so is the trajectory. □

Lemma A.2 (Effective resistance contraction)**.** *For $(i,j) \in \mathcal{E}$ and any $x \in \mathbb{R}^d$, we have:*

$$\|(\mathbf{e}_i - \mathbf{e}_j)(\mathbf{e}_i - \mathbf{e}_j)^\mathsf{T} x\|^2_{\mathbf{\Lambda}^+} \leq \chi_2 \|(\mathbf{e}_i - \mathbf{e}_j)(\mathbf{e}_i - \mathbf{e}_j)^\mathsf{T} x\|^2.$$

Proof. Indeed, we note that:

$$\begin{aligned}
\|(\mathbf{e}_i - \mathbf{e}_j)(\mathbf{e}_i - \mathbf{e}_j)^\mathsf{T} x\|^2_{\mathbf{\Lambda}^+} &= x^\mathsf{T}(\mathbf{e}_i - \mathbf{e}_j)(\mathbf{e}_i - \mathbf{e}_j)^\mathsf{T}\mathbf{\Lambda}^+(\mathbf{e}_i - \mathbf{e}_j)(\mathbf{e}_i - \mathbf{e}_j)^\mathsf{T} x && (10)\\
&\leq 2\chi_2 x^\mathsf{T}(\mathbf{e}_i - \mathbf{e}_j)(\mathbf{e}_i - \mathbf{e}_j)^\mathsf{T} x && (11)\\
&= \chi_2 \|(\mathbf{e}_i - \mathbf{e}_j)(\mathbf{e}_i - \mathbf{e}_j)^\mathsf{T} x\|^2 && (12)
\end{aligned}$$

□

Lemma A.3 (Bound on the resistance)**.**

For $(i,j) \in \mathcal{E}$, $(e_i - e_j)^\mathsf{T}\Lambda^+(e_i - e_j) \leq \frac{1}{\lambda_{ij}+\lambda_{ji}}$.

Proof. For a graph such that $|\bar{\mathcal{E}}| = 1$, the inequality is trivial. Now, we assume that there are other edges than (i,j) or (j,i). Thus, we have:

$$\Lambda \succcurlyeq (\lambda_{ij} + \lambda_{ji})(e_i - e_j)(e_i - e_j)^\mathsf{T} + \tilde{\lambda}\tilde{\pi}$$

where $\tilde{\pi}(e_i - e_j) = 0, \tilde{\pi}\mathbf{1} = 0$, rank$(\tilde{\pi}) = n - 1$, $\tilde{\pi}$ orthogonal projector and $\tilde{\lambda} > 0$ small enough. In this case: $\Big((\lambda_{ij} + \lambda_{ji})(e_i - e_j)(e_i - e_j)^\mathsf{T} + \tilde{\lambda}\tilde{\pi}\Big)^+ \succcurlyeq \Lambda^+$, but this implies that $\frac{1}{\lambda_{ij}+\lambda_{ji}} \geq (e_i - e_j)^\mathsf{T}\Lambda^+(e_i - e_j)$. □

Proof of Lemma 3.3. First, we note that Λ is symmetric and has a non-negative spectrum, as:

$$x^\top \Lambda x = \sum_{(ij)\in\mathcal{E}} \lambda_{ij} \|x_i - x_j\|^2 \,.$$

From this, we also clearly see that $\chi_1 = +\infty$ iff the graph is disconnected. Next, assuming that the graph is connected, 0 is an eigenvalue of Λ with multiplicity 1 and by definition of χ_1, we have $\chi_1 \geq \chi_2$. As we also have the following:

$$\sum_{(i,j)\in\mathcal{E}} \lambda_{ij}(e_i - e_j)^\top \Lambda^+ (e_i - e_j) = \mathrm{Tr}\,(\Lambda^+\Lambda) = n-1\,,$$

we can write:

$$n-1 \leq 2\chi_2 \sum_{(i,j)\in\mathcal{E}} \lambda_{ij} = \chi_2 \mathrm{Tr}\,\Lambda$$

and get $\frac{n-1}{\mathrm{Tr}\,\Lambda} \leq \chi_2$. Finally, note that: $\mathrm{Tr}\,(\Lambda) = 2\sum_{(i,j)\in\mathcal{E}} \lambda_{ij} \leq 2|\bar{\mathcal{E}}| \sup_{(i,j)\in\mathcal{E}}(\lambda_{ij}+\lambda_{ji})$ and $\mathrm{Tr}\,(\Lambda) \leq (n-1)\|\Lambda\|$, so that, using Lemma A.3:

$$\sqrt{\chi_2}\mathrm{Tr}(\Lambda) \leq \frac{1}{\sqrt{2\inf_{(i,j)\in\mathcal{E}}(\lambda_{ij}+\lambda_{ji})}} \sqrt{\mathrm{Tr}\,\Lambda}\sqrt{\mathrm{Tr}\,\Lambda} \tag{13}$$

$$\leq \sqrt{\frac{\mathrm{Tr}\,\Lambda}{2\inf_{(i,j)\in\mathcal{E}}(\lambda_{ij}+\lambda_{ji})}} \sqrt{2|\bar{\mathcal{E}}| \sup_{(i,j)\in\mathcal{E}}(\lambda_{ij}+\lambda_{ji})} \tag{14}$$

$$\leq \sqrt{\kappa}\sqrt{|\bar{\mathcal{E}}|}\sqrt{(n-1)\|\Lambda\|} \tag{15}$$

□

B. Saddle Point Reformulation

With $0 < \nu < \mu$ and introducing an extra dual variable $\hat{x}$, we get:

$$\begin{aligned}
\inf_{x\in\mathbb{R}^d} \sum_{i=1}^n f_i(x) &= \inf_{\substack{x,\hat{x}\in\mathbb{R}^{n\times d}\\ x=\hat{x},\pi\hat{x}=0}} \sum_{i=1}^n f_i(x_i) - \frac{\nu}{2}\|x\|^2 + \frac{\nu}{2}\|\hat{x}\|^2 \\
&= \inf_{x,\hat{x}\in\mathbb{R}^{n\times d}} \sup_{y,z\in\mathbb{R}^{n\times d}} \sum_{i=1}^n f_i(x_i) - \frac{\nu}{2}\|x\|^2 + \frac{\nu}{2}\|\hat{x}\|^2 + \langle y, \hat{x}-x\rangle + \langle z, \pi\hat{x}\rangle \\
&= \inf_{x\in\mathbb{R}^{n\times d}} \sup_{y,z\in\mathbb{R}^{n\times d}} \inf_{\hat{x}\in\mathbb{R}^{n\times d}} \sum_{i=1}^n f_i(x_i) - \frac{\nu}{2}\|x\|^2 + \frac{\nu}{2}\|\hat{x}\|^2 + \langle y, \hat{x}-x\rangle + \langle z, \pi\hat{x}\rangle \\
&= \inf_{x\in\mathbb{R}^{n\times d}} \sup_{y,z\in\mathbb{R}^{n\times d}} \sum_{i=1}^n f_i(x_i) - \frac{\nu}{2}\|x\|^2 - \langle x, y\rangle - \frac{1}{2\nu}\|\pi z + y\|^2 \,.
\end{aligned}$$

C. Proof of the main Theorem (Th. 3.5)

Basic reminds about the Bregman divergence For a smooth convex function F,

$$d_F(x,y) \triangleq F(x) - F(y) - \langle \nabla F(y), x-y\rangle$$

is its Bregman divergence. Next, we set $\nu = \frac{\mu}{2}$ such that, for $F(x) = \sum_{i=1}^n f_i(x_i) - \frac{\nu}{2}\|x\|^2$, then F is L-smooth, ν-strongly convex, and we get:

$$\frac{1}{2L}\|\nabla F(x) - \nabla F(y)\|^2 \leq d_F(x,y) \leq \frac{L}{2}\|x-y\|^2\,,$$

and

$$\frac{\nu}{2}\|x-y\|^2 \leq d_F(x,y) \leq \frac{1}{2\nu}\|\nabla F(x) - \nabla F(y)\|^2\,,$$

Proof of Theorem 3.6. We introduce for a positive semi-definite matrix A, $\|x\|_A^2 \triangleq x^\top A x$. For our proof, we rely on the notation of Eq. 7, and we introduce $X \triangleq (x, \tilde{x}, \tilde{y}), Y \triangleq (y, z, \tilde{z})$ and the following Lyapunov potential:

$$\begin{aligned}\Phi(t, X, Y) \triangleq\ & A_t d_F(x, x^*) + \tilde{A}_t \|\tilde{x} - x^*\|^2 + B_t \|y - y^*\|^2 + \tilde{B}_t \|\tilde{y} - y^*\|^2 \\ & + C_t \|z + y - z^* - y^*\|^2 + \tilde{C}_t \|\tilde{z} - z^*\|^2_{\mathbf{\Lambda}^+}\,,\end{aligned}$$

where $A_t, \tilde{A}_t, B_t, \tilde{B}_t, C_t, \tilde{C}_t$ are non-negative functions to be defined. We will use this potential to control the trajectories.

Because Φ is smooth, the SDE is a smooth trajectory, we get via Ito's formula (Last & Penrose, 2017) applied to the semi-martingale (X_t, Y_t) on any intervals $[0, T]$:

$$\begin{aligned}\Phi(T, X_T, Y_T) = \Phi(0, X_0, Y_0) &+ \int_0^T \langle \nabla \Phi(t, X_t, Y_t), \begin{pmatrix} 1 \\ a_1(X_t, Y_t) \\ a_2(X_t, Y_t) \end{pmatrix} \rangle dt \\ &+ \sum_{i=1}^n \int_0^T \big(\Phi(t, X_t + b_1^i(X_t), Y_t) - \Phi(t, X_t, Y_t)\big) dt \\ &+ \sum_{(i,j)\in\mathcal{E}} \int_0^T \big(\Phi(t, X_t, Y_t + b_2^{ij}(Y_t)) - \Phi(t, X_t, Y_t)\big) \lambda_{ij} dt \\ &+ \Theta_T\,,\end{aligned}$$

where the following quantity is a Martingale:

$$\begin{aligned}\Theta_u \triangleq & \sum_{i=1}^n \int_0^u \big(\Phi(t, X_{t^-}, Y_{t^-} + b_1^i(X_{t^-})) - \Phi(u, X_{t^-}, Y_{t^-})\big)(dN_i(t) - dt) \\ & + \sum_{(i,j)\in\mathcal{E}} \int_0^u \big(\Phi(t, X_{t^-} + b_2^{ij}(X_{t^-}), Y_{t^-}) - \Phi(t, X_{t^-}, Y_{t^-})\big)(dM_{ij}(t) - \lambda_{ij} dt)\,.\end{aligned}$$

Taking the expectation, we get that, as the initialization is deterministic:

$$\begin{aligned}\mathbb{E}[\Phi(T, X_T, Y_T)] = \Phi(0, X_0, Y_0) &+ \int_0^T \langle \nabla \Phi(t, X_t, Y_t), \begin{pmatrix} 1 \\ a_1(X_t, Y_t) \\ a_2(X_t, Y_t) \end{pmatrix} \rangle dt \\ &+ \sum_{i=1}^n \int_0^T \big(\Phi(t, X_t + b_1^i(X_t), Y_t) - \Phi(t, X_t, Y_t)\big) dt \\ &+ \sum_{(i,j)\in\mathcal{E}} \int_0^T \big(\Phi(t, X_t, Y_t + b_2^{ij}(Y_t)) - \Phi(t, X_t, Y_t)\big) \lambda_{ij} dt\,,\end{aligned}$$

To show that the integrand term is negative, we will use the following technical Lemma, which is also difficult to prove and whose proof is deferred to Appendix C.1:

Lemma C.1. *If:*

$$\begin{array}{llllll} \eta = \frac{1}{8}\sqrt{\frac{\nu}{L}} & \gamma = \frac{1}{4L} & \delta = \frac{1}{4}\sqrt{\frac{\nu}{L}} & \alpha = \frac{1}{4}\sqrt{\frac{\nu}{L}} & \beta = \frac{1}{2} & \theta = \frac{1}{2}\sqrt{\frac{L}{\nu}} \\ \tilde{\eta} = \frac{1}{8}\sqrt{\frac{\nu}{L}} & \tilde{\gamma} = \frac{1}{4\sqrt{\nu L}} & \tilde{\delta} = 1 & \tilde{\alpha} = \frac{1}{8}\sqrt{\frac{\nu}{L}} & \tilde{\beta} = 2\chi_1[\Lambda]\sqrt{\frac{L}{\nu}} & \nu = \frac{\mu}{2} \end{array}$$

and

$$\begin{array}{lll} A_t = e^{\frac{t}{8\sqrt{2}}\sqrt{\frac{\mu}{L}}} & \tilde{A}_t = \frac{\mu}{8} e^{\frac{t}{8\sqrt{2}}\sqrt{\frac{\mu}{L}}} & \tilde{B}_t = \frac{1}{8L} e^{\frac{t}{8\sqrt{2}}\sqrt{\frac{\mu}{L}}} \\ B_t = \frac{1}{16L} e^{\frac{t}{8\sqrt{2}}\sqrt{\frac{\mu}{L}}} & C_t = \frac{1}{2\mu} e^{\frac{t}{8\sqrt{2}}\sqrt{\frac{\mu}{L}}} & \tilde{C}_t = \frac{1}{32\chi_1 L} e^{\frac{t}{8\sqrt{2}}\sqrt{\frac{\mu}{L}}}\,. \end{array}$$

then:

$$\langle \nabla\Phi(t,X_t,Y_t), \begin{pmatrix} 1 \\ a_1(X_t,Y_t) \\ a_2(X_t,Y_t) \end{pmatrix} \rangle + \big(\Phi(t,X_t+b_1(X_t),Y_t) - \Phi(t,X_t,Y_t)\big) + \sum_{(i,j)\in\mathcal{E}} \lambda_{ij}\big(\Phi(t,X_t,Y_t+b_2^{ij}(Y_t)) - \Phi(t,X_t,Y_t)\big) \leq 0 \; a.s. \, .$$

Now, we remark that if we have $F(x) = f(x) - \frac{\mu}{4}\|x\|^2$, and initialize with $\tilde{x}_0 = x_0$, $y_0 = \tilde{y}_0 = \nabla F(x_0)$ and $z_0 = \tilde{z}_0 = -\pi\nabla F(x_0)$, then, given the linear relation between $A_t, \tilde{A}_t, B_t, \tilde{B}_t, C_t, \tilde{C}_t$, the L smoothness and the fact π is an orthogonal projection, we get:

$$\begin{aligned}
\Phi(0,X_0,Y_0) &\leq d_F(x_0,x^*) + \frac{\mu}{8}\|x_0-x^*\|^2 + \frac{1}{16L}\|\nabla F(x_0)-\nabla F(x^*)\|^2 \\
&+ \frac{1}{8L}\|\nabla F(x_0)-\nabla F(x^*)\|^2 + \frac{1}{2\mu}\|(\mathbf{I}-\pi)(\nabla F(x_0)-\nabla F(x^*))\|^2 \\
&+ \frac{1}{32}\frac{\chi_1}{L\chi_1}\|\pi(\nabla F(x_0)-\nabla F(x^*))\|^2 \\
&\leq (\frac{L}{2}+\frac{\mu}{8}+\frac{L}{16}+\frac{L}{8}+\frac{L^2}{2\mu}+\frac{L}{32})\|x_0-x^*\|^2
\end{aligned}$$

In particular, as $\frac{\mu}{4}\|x-x^*\|^2 \leq d_F(x,x^*)$ it implies that:

$$\mathbb{E}[\|x_t-x^*\|^2] \leq (\frac{1}{2}+\frac{23}{8}\frac{L}{\mu}+2\frac{L^2}{\mu^2})\|x_0-x^*\|^2 e^{-\frac{t}{8\sqrt{2}}\sqrt{\frac{\mu}{L}}}$$

Finally, we note that the expected number of gradients between $[0,T]$ is given by:

$$\mathbb{E}[\sum_{i=1}^n N_i(T)] = nT\,,$$

and similarly, the number of edges activated is given by:

$$\mathbb{E}[\sum_{1\leq i,j\leq n} M_{ij}(T)] = \sum_{1\leq i,j\leq n}\int_0^T \lambda_{ij}\,dt = \frac{T}{2}\mathrm{Tr}\,\Lambda.$$

□

C.1. Proof of the Lemma C.1

We first state a couple of inequalities that we will combine to obtain a bound on our Lyapunov function. In all this section, for a given variable x, we denote by x^+ the value of x right after a Poisson update.

Lemma C.2. *First:*

$$\begin{aligned}
\phi_A &\triangleq A_t(d_F(x^+,x^*)-d_F(x,x^*)) + \tilde{A}_t(\|\tilde{x}^+-x^*\|^2-\|\tilde{x}-x^*\|^2) \\
&+ \eta A_t\langle\tilde{x}-x,\nabla F(x)-\nabla F(x^*)\rangle + 2\tilde{\eta}\tilde{A}_t\langle x-\tilde{x},\tilde{x}-x^*\rangle \qquad (16)\\
&\leq \|\nabla F(x)-\tilde{y}\|^2\left(A_t\frac{L\gamma^2}{2}-A_t\gamma+\tilde{A}_t\tilde{\gamma}^2\right) \\
&+ A_t\gamma\langle\nabla F(x)-\tilde{y},y^*-\tilde{y}\rangle + 2\tilde{\gamma}\tilde{A}_t\langle\tilde{y}-y^*,\tilde{x}-x^*\rangle \qquad (17)\\
&- 2\tilde{\gamma}\tilde{A}_t\left(d_F(\tilde{x},x^*)+d_F(x^*,x)-d_F(\tilde{x},x)\right) \\
&- \eta A_t(d_F(\tilde{x},x)+d_F(x,x^*)-d_F(\tilde{x},x^*)) - \tilde{A}_t\tilde{\eta}\|\tilde{x}-x^*\|^2 + \tilde{A}_t\tilde{\eta}\|x-x^*\|^2
\end{aligned}$$

Proof. First, we have to use optimality conditions and smoothness, as well as the separability of F:

$$d_F(x^+, x^*) - d_F(x, x^*) = d_F(x^+, x) - \langle x^+ - x, \nabla F(x^*) - \nabla F(x)\rangle \tag{18}$$
$$\leq \frac{L}{2}\|x^+ - x\|^2 - \langle x^+ - x, \nabla F(x^*) - \nabla F(x)\rangle \tag{19}$$
$$= \frac{L\gamma^2}{2}\|\tilde{y} - \nabla F(x)\|^2 - \gamma\|\nabla F(x) - \tilde{y}\|^2 + \gamma\langle \nabla F(x) - \tilde{y}, y^* - \tilde{y}\rangle \tag{20}$$

Next, we note that, again using optimality conditions:

$$\|\tilde{x}^+ - x^*\|^2 - \|\tilde{x} - x^*\|^2 = 2\langle \tilde{x}^+ - \tilde{x}, \tilde{x} - x^*\rangle + \|\tilde{x}^+ - \tilde{x}\|^2 \tag{21}$$
$$= -2\tilde{\gamma}\langle \nabla F(x) - \tilde{y}, \tilde{x} - x^*\rangle + \tilde{\gamma}^2\|\nabla F(x) - \tilde{y}\|^2 \tag{22}$$
$$= -2\tilde{\gamma}\langle \nabla F(x) - \nabla F(x^*), \tilde{x} - x^*\rangle + 2\tilde{\gamma}\langle \tilde{y} - y^*, \tilde{x} - x^*\rangle + \tilde{\gamma}^2\|\nabla F(x) - \tilde{y}\|^2 \tag{23}$$
$$= -2\tilde{\gamma}(d_F(\tilde{x}, x^*) + d_F(x^*, x) - d_F(\tilde{x}, x)) + 2\tilde{\gamma}\langle \tilde{y} - y^*, \tilde{x} - x^*\rangle + \tilde{\gamma}^2\|\nabla F(x) - \tilde{y}\|^2 \tag{24}$$

Momentum in x associated with the term $d_F(x, x^*)$ gives:

$$\eta\langle \tilde{x} - x, \nabla F(x) - \nabla F(x^*)\rangle = -\eta(d_F(\tilde{x}, x) + d_F(x, x^*) - d_F(\tilde{x}, x^*)) \tag{25}$$

and momentum in $\tilde{x}$ associated with $\|\tilde{x} - x^*\|^2$ leads to:

$$2\tilde{\eta}\langle x - \tilde{x}, \tilde{x} - x^*\rangle = -2\tilde{\eta}\|\tilde{x} - x^*\|^2 + 2\tilde{\eta}\langle x - x^*, \tilde{x} - x^*\rangle \leq -\tilde{\eta}\|\tilde{x} - x^*\|^2 + \tilde{\eta}\|x - x^*\|^2 \tag{26}$$

□

Lemma C.3. *Next, we show that if* $\alpha B_t = \frac{\delta}{2}\tilde{B}_t$*:*

$$\phi_B \triangleq B_t(\|y^+ - y^*\|^2 - \|y - y^*\|^2) + \tilde{B}_t(\|\tilde{y}^+ - y^*\|^2 - \|\tilde{y} - y^*\|^2) + 2\alpha B_t\langle y - y^*, \tilde{y} - y\rangle - 2\theta\tilde{B}_t\langle y + z + \nu\tilde{x}, \tilde{y} - y^*\rangle + 2\alpha C_t\langle \tilde{y} - y, z + y - y^* - z^*\rangle \tag{27}$$
$$\leq -\frac{\delta}{2}\tilde{B}_t\|\tilde{y} - y^*\|^2 - \frac{\delta}{2}\tilde{B}_t\|y - y^*\|^2 - 2\tilde{\delta}\tilde{B}_t\langle \nabla F(x) - \tilde{y}, y^* - \tilde{y}\rangle + \delta\tilde{B}_t\|\nabla F(x) - \nabla F(x^*)\|^2 + \left((\delta + \tilde{\delta})^2 - \delta\right)\tilde{B}_t\|\nabla F(x) - y\|^2 - 2\theta\tilde{B}_t\langle y + z - y^* - z^*, \tilde{y} - y^*\rangle - 2\theta\nu\tilde{B}_t\langle \tilde{x} - x^*, \tilde{y} - y^*\rangle + 2\alpha C_t\langle \tilde{y} - y, z + y - y^* - z^*\rangle \tag{28}$$

Proof. Using optimality conditions:

$$\|\tilde{y}^+ - y^*\|^2 - \|\tilde{y} - y^*\|^2 = 2\langle \tilde{y} - y^*, \tilde{y}^+ - \tilde{y}\rangle + \|\tilde{y}^+ - \tilde{y}\|^2 \tag{29}$$
$$= 2\delta\langle \nabla F(x) - \tilde{y}, \tilde{y} - y^*\rangle + 2\tilde{\delta}\langle \nabla F(x) - \tilde{y}, \tilde{y} - y^*\rangle + (\delta + \tilde{\delta})^2\|\nabla F(x) - \tilde{y}\|^2 \tag{30}$$
$$= -2\tilde{\delta}\langle \nabla F(x) - \tilde{y}, y^* - \tilde{y}\rangle + \delta\|\nabla F(x) - \nabla F(x^*)\|^2 - \delta\|\tilde{y} - y^*\|^2 + \left((\delta + \tilde{\delta})^2 - \delta\right)\|\nabla F(x) - \tilde{y}\|^2 \tag{31}$$

The momentum in $\tilde{y}$ associated with the term $\|\tilde{y} - y^*\|^2$ gives:

$$-2\theta\tilde{B}_t\langle y + z + \nu\tilde{x}, \tilde{y} - y^*\rangle = -2\theta\tilde{B}_t\langle y + z - y^* - z^*, \tilde{y} - y^*\rangle - 2\theta\nu\tilde{B}_t\langle \tilde{x} - x^*, \tilde{y} - y^*\rangle \tag{32}$$

The momentum in y associated with the term $\|y - y^*\|^2$ gives:

$$2\alpha B_t\langle \tilde{y} - y, y - y^*\rangle = -\alpha B_t\|y - y^*\|^2 - \alpha B_t\|\tilde{y} - y\|^2 + \alpha B_t\|\tilde{y} - y^*\|^2 \tag{33}$$

and the one associated with $\|y + z - y^* - z^*\|^2$:

$$2\alpha C_t\langle \tilde{y} - y, z + y - y^* - z^*\rangle \tag{34}$$

□

Lemma C.4. *Finally, assuming $\theta\tilde{B}_t = \tilde{\beta}\tilde{C}_t = \alpha C_t$, letting $1 \geq \tilde{\tau} > 0$, $z_{ij}^+ = z - \beta(\mathbf{e}_i - \mathbf{e}_j)(\mathbf{e}_i - \mathbf{e}_j)^\mathsf{T}(y+z)$ and $\tilde{z}_{ij}^+ = \tilde{z} - \tilde{\beta}(\mathbf{e}_i - \mathbf{e}_j)(\mathbf{e}_i - \mathbf{e}_j)^\mathsf{T}(y+z)$:*

$$\begin{aligned}
&\phi_C - 2\theta\tilde{B}_t\langle y+z-y^*-z^*, \tilde{y}-y^*\rangle + 2\alpha C_t\langle \tilde{y}-y, z+y-y^*-z^*\rangle \triangleq \\
&\quad \sum_{ij}\lambda_{ij}C_t\Big(\|y+z_{ij}^+-y^*-z^*\|^2 - \|y+z-y^*-z^*\|^2\Big) \\
&\quad + \sum_{ij}\lambda_{ij}\tilde{C}_t\Big(\|\tilde{z}_{ij}^+-z^*\|^2_{\mathbf{\Lambda}^+} - \|\tilde{z}-z^*\|^2_{\mathbf{\Lambda}^+}\Big) + 2\tilde{\alpha}\tilde{C}_t\langle z-\tilde{z}, \tilde{z}-z^*\rangle_{\mathbf{\Lambda}^+} \qquad (35)\\
&\quad + 2\alpha C_t\langle \tilde{z}+\tilde{y}-z-y, z+y-y^*-z^*\rangle - 2\theta\tilde{B}_t\langle y+z-y^*-z^*, \tilde{y}-y^*\rangle \\
&\quad \leq \tilde{\beta}^2\chi_2\tilde{C}_t\sum_{(i,j)\in\mathcal{E}}\lambda_{ij}\|(\mathbf{e}_i-\mathbf{e}_j)(\mathbf{e}_i-\mathbf{e}_j)^\mathsf{T}(y+z)\|^2 \\
&\quad + \beta(\beta-1)C_t\sum_{(i,j)\in\mathcal{E}}\lambda_{ij}\|(\mathbf{e}_i-\mathbf{e}_j)(\mathbf{e}_i-\mathbf{e}_j)^\mathsf{T}(y+z)\|^2 \\
&\quad - \alpha C_t\|y+z-y^*-z^*\|^2 + \tilde{\alpha}\chi_1\tilde{C}_t\|z-z^*\|^2 - \tilde{\alpha}\tilde{C}_t\|\tilde{z}-z^*\|^2_{\mathbf{\Lambda}^+} \\
&\quad - \tilde{\tau}\frac{1}{2}\tilde{\beta}\frac{\nu}{L}\tilde{C}_t\|z-z^*\|^2 + \tilde{\tau}\frac{\nu}{L}\frac{2\alpha\theta}{\delta}B_t\|y-y^*\|^2 \qquad (36)
\end{aligned}$$

Proof. Having in mind that $\pi(y^*+z^*) = 0$ and $\mathbf{\Lambda}^+\mathbf{\Lambda} = \pi$, we get, using Lemma A.1 and Lemma A.2 on the inequality (40):

$$\begin{aligned}
\Delta_{\tilde{z}} &\triangleq \sum_{(i,j)\in\mathcal{E}}\lambda_{ij}\big(\|\tilde{z}_{ij}^+-z^*\|^2_{\mathbf{\Lambda}^+} - \|\tilde{z}-z^*\|^2_{\mathbf{\Lambda}^+}\big) \qquad (37)\\
&= \sum_{(i,j)\in\mathcal{E}}\lambda_{ij}2\langle \tilde{z}-z^*, \tilde{z}_{ij}^+-\tilde{z}\rangle_{\mathbf{\Lambda}^+} + \|\tilde{z}_{ij}^+-\tilde{z}\|^2_{\mathbf{\Lambda}^+} \qquad (38)\\
&= -2\tilde{\beta}\sum_{(i,j)\in\mathcal{E}}\lambda_{ij}\langle \tilde{z}-z^*, (\mathbf{e}_i-\mathbf{e}_j)(\mathbf{e}_i-\mathbf{e}_j)^\mathsf{T}(y+z-y^*-z^*)\rangle_{\mathbf{\Lambda}^+} \\
&\quad + \sum_{(i,j)\in\mathcal{E}}\lambda_{ij}\tilde{\beta}^2\|(\mathbf{e}_i-\mathbf{e}_j)(\mathbf{e}_i-\mathbf{e}_j)^\mathsf{T}(y+z)\|^2_{\mathbf{\Lambda}^+} \qquad (39)\\
&\leq -2\tilde{\beta}\langle \tilde{z}-z^*, \mathbf{\Lambda}^+\mathbf{\Lambda}(y+z)\rangle + \chi_2\tilde{\beta}^2\sum_{(i,j)\in\mathcal{E}}\lambda_{ij}\|(\mathbf{e}_i-\mathbf{e}_j)(\mathbf{e}_i-\mathbf{e}_j)^\mathsf{T}(y+z)\|^2 \qquad (40)\\
&= -2\tilde{\beta}\langle \tilde{z}-z^*, \pi(y+z)\rangle + \chi_2\tilde{\beta}^2\sum_{(i,j)\in\mathcal{E}}\lambda_{ij}\|(\mathbf{e}_i-\mathbf{e}_j)(\mathbf{e}_i-\mathbf{e}_j)^\mathsf{T}(y+z)\|^2 \qquad (41)
\end{aligned}$$

We also have, noting that y^+, the value of y after a Poisson update, is equal to y:

$$\begin{aligned}
\Delta_z &\triangleq \sum_{(i,j)\in\mathcal{E}}\lambda_{ij}(\|y^+ + z_{ij}^+ - y^* - z^*\|^2 - \|y+z-y^*-z^*\|^2) \qquad (42)\\
&= 2\sum_{(i,j)\in\mathcal{E}}\lambda_{ij}\langle y+z_{ij}^+-y-z, y+z-y^*-z^*\rangle + \sum_{(i,j)\in\mathcal{E}}\lambda_{ij}\|y+z_{ij}^+-y-z\|^2 \qquad (43)\\
&= -2\sum_{(i,j)\in\mathcal{E}}\beta\lambda_{ij}\langle(\mathbf{e}_i-\mathbf{e}_j)(\mathbf{e}_i-\mathbf{e}_j)^\mathsf{T}(y+z), y+z-y^*-z^*\rangle \\
&\quad + \sum_{(i,j)\in\mathcal{E}}\beta^2\lambda_{ij}\|(\mathbf{e}_i-\mathbf{e}_j)(\mathbf{e}_i-\mathbf{e}_j)^\mathsf{T}(y+z)\|^2 \qquad (44)\\
&= \beta(\beta-1)\sum_{(i,j)\in\mathcal{E}}\lambda_{ij}\|(\mathbf{e}_i-\mathbf{e}_j)(\mathbf{e}_i-\mathbf{e}_j)^\mathsf{T}(y+z)\|^2 \qquad (45)
\end{aligned}$$

The momentum in $\tilde{z}$ associated with $\|\tilde{z}-z^*\|^2_{\mathbf{\Lambda}^+}$ gives:

$$2\tilde{\alpha}\tilde{C}_t\langle z-\tilde{z}, \tilde{z}-z^*\rangle_{\mathbf{\Lambda}^+} \leq \tilde{\alpha}\chi_1\tilde{C}_t\|z-z^*\|^2 - \tilde{\alpha}\tilde{C}_t\|\tilde{z}-z^*\|^2_{\mathbf{\Lambda}^+} \qquad (46)$$

And the one in z associated with $\|y+z-y^*-z^*\|^2$ gives:

$$2\alpha C_t\langle \tilde{z}-z, z+y-y^*-z^*\rangle \tag{47}$$

Then, assuming that $\theta\tilde{B}_t = \tilde{\beta}\tilde{C}_t = \alpha C_t$, we have:

$$\begin{aligned}
&2\alpha C_t\langle \tilde{y}-y, z+y-y^*-z^*\rangle - 2\tilde{\beta}\tilde{C}_t\langle \tilde{z}-z^*, y+z-y^*-z^*\rangle \\
&- 2\theta\tilde{B}_t\langle y+z-y^*-z^*, \tilde{y}-y^*\rangle + 2\alpha C_t\langle \tilde{z}-z, z+y-y^*-z^*\rangle && (48)\\
&= -2\alpha C_t\|y+z-y^*-z^*\|^2 && (49)
\end{aligned}$$

At this stage, we split the negative term (49) into two halves, upper-bounding one of the halves by remembering that $\frac{\nu}{L}\leq 1$ and introducing $1\geq\tilde{\tau}>0$:

$$\begin{aligned}
-\alpha C_t\|y+z-y^*-z^*\|^2 &\leq -\tilde{\tau}\frac{\nu}{L}\alpha C_t\|y+z-y^*-z^*\|^2 && (50)\\
&= -\tilde{\tau}\tilde{\beta}\frac{\nu}{L}\tilde{C}_t\|y+z-y^*-z^*\|^2 && (51)\\
&\leq -\tilde{\tau}\frac{1}{2}\tilde{\beta}\frac{\nu}{L}\tilde{C}_t\|z-z^*\|^2 + \tilde{\tau}\tilde{\beta}\frac{\nu}{L}\tilde{C}_t\|y-y^*\|^2 && (52)\\
&= -\tilde{\tau}\frac{1}{2}\tilde{\beta}\frac{\nu}{L}\tilde{C}_t\|z-z^*\|^2 + \tilde{\tau}\frac{\nu}{L}\frac{2\alpha\theta}{\delta}B_t\|y-y^*\|^2 && (53)
\end{aligned}$$

□

Keeping in mind that $\theta\tilde{B}_t = \tilde{\beta}\tilde{C}_t = \alpha C_t$ and $\frac{\delta}{2}\tilde{B}_t = \alpha B_t$, we put everything together. Defining $\Psi = \frac{\partial\Phi}{\partial t} + \phi_A + \phi_B + \phi_C$, we have:

$$\begin{aligned}
\Psi \leq\ & \|\nabla F(x)-\tilde{y}\|^2\left(A_t\frac{L\gamma^2}{2} - A_t\gamma + \tilde{A}_t\tilde{\gamma}^2 + \left((\delta+\tilde{\delta})^2-\delta\right)\tilde{B}_t\right) && (54)\\
&+ \|\tilde{z}-z^*\|^2_{\mathbf{\Lambda}^+}\left(-\tilde{\alpha}\tilde{C}_t + \tilde{C}'_t\right) && (55)\\
&+ \|\tilde{y}-y^*\|^2(\tilde{B}'_t - \frac{\delta}{2}\tilde{B}_t) && (56)\\
&+ \|x-x^*\|^2(\tilde{A}_t\tilde{\eta} - \tilde{A}_t\frac{\nu\tilde{\gamma}}{2}) && (57)\\
&+ \|\tilde{x}-x^*\|^2(\tilde{A}'_t - \tilde{A}_t\tilde{\eta}) && (58)\\
&+ \|\nabla F(x)-\nabla F(x^*)\|^2(\delta\tilde{B}_t - \frac{\tilde{\gamma}}{2L}\tilde{A}_t) && (59)\\
&+ \sum_{(i,j)\in\mathcal{E}}\lambda_{ij}\|(\mathbf{e}_i-\mathbf{e}_j)(\mathbf{e}_i-\mathbf{e}_j)^\mathsf{T}(y+z)\|^2\left(\chi_2\tilde{\beta}^2\tilde{C}_t + \beta(\beta-1)C_t\right) && (60)\\
&+ \|z-z^*\|^2(\chi_1\tilde{\alpha} - \tilde{\tau}\frac{1}{2}\tilde{\beta}\frac{\nu}{L})\tilde{C}_t && (61)\\
&+ \|y-y^*\|(B'_t - (1-\tilde{\tau}\frac{\nu}{L}\frac{2\theta}{\delta})\alpha B_t) && (62)\\
&+ \|y+z-y^*-z^*\|^2(C'_t - \alpha C_t) && (63)\\
&+ d_F(x,x^*)(A'_t - \eta A_t) && (64)\\
&+ d_F(\tilde{x},x)(-A_t\eta + 2\tilde{\gamma}\tilde{A}_t) && (65)\\
&+ d_F(\tilde{x},x^*)(A_t\eta - 2\tilde{\gamma}\tilde{A}_t) && (66)\\
&+ \langle\nabla F(x)-\tilde{y}, y^*-\tilde{y}\rangle(-2\tilde{\delta}\tilde{B}_t + \gamma A_t) && (67)\\
&+ \langle\tilde{y}-y^*, \tilde{x}-x^*\rangle\left(2\tilde{\gamma}\tilde{A}_t - 2\theta\nu\tilde{B}_t\right) && (68)
\end{aligned}$$

Resolution

Proof of Lemma C.1. Our goal is to put to zero all of the terms appearing next to scalar products and make the factors of positive quantities (norms or divergences) less or equal to zero. Given our relations, we guess that each exponential has the same rate. Thus, with $\tau > 0$, we fix $\frac{\delta}{2} = \tilde{\eta} = \eta = \tilde{\alpha} = \tau\sqrt{\frac{\nu}{L}}$, which leads to $\tilde{\gamma} = \frac{2\tau}{\sqrt{\nu L}}$ using Eq. (57). Also, from Eq. (66):

$$4\tilde{A}_t = \nu A_t.$$

Next, from Eq. (59) and Eq. (68), it's necessary that:

$$2L\delta = \theta\nu\,,$$

thus $\theta = 4\tau\sqrt{\frac{L}{\nu}}$. From Eq. (68), we get:

$$\tilde{A}_t = 2L\nu\tilde{B}_t.$$

Combining this previous equation with Eq. (67), as $4\tilde{A}_t = \nu A_t$, we have $\tilde{\delta} = 4L\gamma$. Next, Eq. (54) gives, with the equations above:

$$\begin{aligned}
A_t(\frac{L\gamma^2}{2} - \gamma) + \tilde{A}_t\tilde{\gamma}^2 + \left((\delta+\tilde{\delta})^2 - \delta\right)\tilde{B}_t &= A_t\frac{L\gamma^2}{2} - A_t\gamma + \frac{\nu}{4}\tilde{\gamma}^2 A_t + \left(\delta^2 + \tilde{\delta}^2 + \delta\right)\frac{A_t}{8L} \\
&= A_t\left(\frac{L\gamma^2}{2} - \gamma + \frac{\nu}{4}\frac{4\tau^2}{\nu L}\right) + A_t(2\tau\sqrt{\frac{\nu}{L}} + 4\tau^2\frac{\nu}{L} + 16L^2\gamma^2)\frac{1}{8L} \\
&\leq A_t(\gamma^2\frac{5}{2}L - \gamma + \frac{5}{4}\frac{\tau^2}{L} + \frac{\sqrt{2}}{8}\frac{\tau}{L})
\end{aligned}$$

We thus pick $\gamma = \frac{1}{4L}$ and $\tau = \frac{1}{8}$, so that $\tilde{\delta} = 1$. Via Eq. (62), we fix $\tilde{\tau} = \frac{1}{8} < 1$. With Eq. (61), we then get:

$$\tilde{\beta} = 2\chi_1\sqrt{\frac{L}{\nu}}$$

We also put $\alpha = 2\tau\sqrt{\frac{\nu}{L}}$ and only one last equation, Eq. (60), needs to be satisfied, for which we pick $\beta = \frac{1}{2}$:

$$\chi_2\tilde{\beta}^2\tilde{C}_t + \beta(\beta-1)C_t = (\chi_2\tilde{\beta}\alpha - \frac{1}{4})C_t$$

This implies that $\chi_2\chi_1 \leq \frac{1}{2}$. In summary, we set:

$$\begin{array}{llllllll}
\eta = \frac{1}{8}\sqrt{\frac{\nu}{L}} & \gamma = \frac{1}{4L} & \delta = \frac{1}{4}\sqrt{\frac{\nu}{L}} & \alpha = \frac{1}{4}\sqrt{\frac{\nu}{L}} & \beta = \frac{1}{2} & \theta = \frac{1}{2}\sqrt{\frac{L}{\nu}} & \tilde{\tau} = \frac{1}{8} \\
\tilde{\eta} = \frac{1}{8}\sqrt{\frac{\nu}{L}} & \tilde{\gamma} = \frac{1}{4\sqrt{\nu L}} & \tilde{\delta} = 1 & \tilde{\alpha} = \frac{1}{8}\sqrt{\frac{\nu}{L}} & \tilde{\beta} = 2\chi_1\sqrt{\frac{L}{\nu}} & \nu = \frac{\mu}{2} & \tau = \frac{1}{8}
\end{array}$$

Now, let's pick: $A_t = e^{t\tau\sqrt{\frac{\nu}{L}}} = e^{\frac{t}{8\sqrt{2}}\sqrt{\frac{\mu}{L}}}$ so that:

$$\begin{array}{lll}
A_t = e^{\frac{t}{8\sqrt{2}}\sqrt{\frac{\mu}{L}}} & \tilde{A}_t = \frac{\mu}{8}e^{\frac{t}{8\sqrt{2}}\sqrt{\frac{\mu}{L}}} & \tilde{B}_t = \frac{1}{8L}e^{\frac{t}{8\sqrt{2}}\sqrt{\frac{\mu}{L}}} \\
B_t = \frac{1}{16L}e^{\frac{t}{8\sqrt{2}}\sqrt{\frac{\mu}{L}}} & C_t = \frac{1}{2\mu}e^{\frac{t}{8\sqrt{2}}\sqrt{\frac{\mu}{L}}} & \tilde{C}_t = \frac{1}{32\chi_1 L}e^{\frac{t}{8\sqrt{2}}\sqrt{\frac{\mu}{L}}}\,.
\end{array}$$

This implies that $\Psi \leq 0$.

□

D. Comparison of complexities with related work

Proof of Prop. 3.9. We consider the settings of concurrent works and, given any gossip matrix admissible for them, we show that DADAO has better rates.

Comparison with ADOM+ (Kovalev et al., 2021a). The ADOM+ setting is:

- *Gossip matrices*: Laplacians $\mathcal{L}$ with $\|\mathcal{L}x - x\|^2 \leq (1 - \frac{1}{\chi})\|x\|^2$ for $\chi \geq 1$ and $x \in \mathbf{1}^\perp$.
- *Total number of gradients to reach ϵ precision*: $\mathcal{O}(n\sqrt{\frac{L}{\mu}}\log\frac{1}{\epsilon})$.
- *Total number of edges activated to reach ϵ precision*: $\mathcal{O}(|\mathcal{E}|\chi\sqrt{\frac{L}{\mu}}\log\frac{1}{\epsilon})$.

If we take an eigenvector of $\mathcal{L}$ for the eigenvalue $\frac{1}{\chi_1[\mathcal{L}]}$, then the Laplacian inequality directly leads to $\left(1 - \frac{1}{\chi_1[\mathcal{L}]}\right)^2 \leq 1 - \frac{1}{\chi}$ and we have:

$$\frac{1}{\chi_1[\mathcal{L}]}\left(\frac{1}{\chi_1[\mathcal{L}]} - 2\right) \leq -\frac{1}{\chi}\,,$$

leading to:

$$\frac{2}{\chi_1[\mathcal{L}]} \geq \frac{1}{\chi_1[\mathcal{L}]}\left(2 - \frac{1}{\chi_1[\mathcal{L}]}\right) \geq \frac{1}{\chi}\,.$$

Thus, $\chi_1[\mathcal{L}] \leq 2\chi$. Remind that, by definition, $\chi_2 \leq \chi_1$. Note that in ADOM+, we also have $\|\mathcal{L}\| \leq 2$, so that $\mathrm{Tr}(\mathcal{L}) \leq 2n$. Then, for any Laplacian matrix $\mathcal{L}$ valid for ADOM+, we consider for DADAO a Λ defined as followed:

$$\Lambda = \sqrt{2\chi_1[\mathcal{L}]\chi_2[\mathcal{L}]}\mathcal{L}$$

Then, DADAO has the same gradient complexity as ADOM+, but the number of communications of DADAO is:

$$\frac{T}{2}\mathrm{Tr}\,\Lambda \leq \frac{1}{2}\sqrt{2\chi_1[\mathcal{L}]\chi_2[\mathcal{L}]}\sqrt{\frac{L}{\mu}}\log\left(\frac{1}{\epsilon}\right)2n = \mathcal{O}(|\mathcal{E}|\chi\sqrt{\frac{L}{\mu}}\log\frac{1}{\epsilon})$$

Consequently, DADAO is better than ADOM+ for all valid configurations of ADOM+ (in the fixed graph topology setting) and DADAO. We note that for the complete graph, $\chi = \mathcal{O}(1)$ and $|\mathcal{E}| = \mathcal{O}(n^2)$, whereas $n\sqrt{2\chi_1[\mathcal{L}]\chi_2[\mathcal{L}]} = \mathcal{O}(n)$ and DADAO has strictly better communication rates than ADOM+ on this graph.

Comparison with Gradient Tracking methods AGT, OGT (Li & Lin, 2021; Song et al., 2021). The setting is described by:

- *Gossip matrices*: Any matrix $\mathcal{L} = \mathbf{I} - W$ with W symmetric doubly-stochastic, *i.e.* such that $\sum_i W_{ij} = 1$ and $\sum_j W_{ij} = 1$.
- *Total number of gradients to reach ϵ precision*: $\mathcal{O}(n\sqrt{\frac{L}{\mu}}\log\frac{1}{\epsilon})$.
- *Total number of edges activated to reach ϵ precision*: $\mathcal{O}(|\mathcal{E}|\frac{1}{\sqrt{\theta}}\sqrt{\frac{L}{\mu}}\log\frac{1}{\epsilon})$, where $\theta = 1 - \|W - \frac{1}{n}\mathbf{1}\mathbf{1}^\mathsf{T}\|$.

If $\kappa = \mathcal{O}(\frac{|\mathcal{E}|}{n})$, using Lemma 3.3, we have:

$$\frac{1}{2}\sqrt{2\chi_1[\mathcal{L}]\chi_2[\mathcal{L}]}\mathrm{Tr}\,\mathcal{L} = \mathcal{O}(|\mathcal{E}|\sqrt{\rho[\mathcal{L}]})$$

Furthermore, as W is stochastic, $\|\mathcal{L}\| \leq 2$ and $\theta \leq \frac{1}{\chi_1[\mathcal{L}]}$, leading to $\rho \leq \frac{2}{\theta}$. Consequently, the communication complexity of DADAO run for $\mathcal{O}(\sqrt{\frac{L}{\mu}}\log\frac{1}{\epsilon})$ iterations recovers the rate of GT. Furthermore, for the complete graph, for any Laplacian $\mathcal{L}$ with $\rho[\mathcal{L}] = \mathcal{O}(1)$ (remind that, by definition $\rho \geq 1$), we have $\sqrt{2\chi_1[\mathcal{L}]\chi_2[\mathcal{L}]}\mathrm{Tr}\,\mathcal{L} = \mathcal{O}(n)$ whereas $|\mathcal{E}| = \mathcal{O}(n^2)$, thus DADAO uses an order of magnitude less communications than GT need to.

Note that for the Star-graph, there is no admissible Laplacian in the framework of (Song et al., 2021).

Comparison with MSDA (Scaman et al., 2017). The setting is described by:

- *Gossip matrices*: any Laplacians admissible for DADAO.
- *Total number of gradients to reach ϵ precision*: $\mathcal{O}(n\sqrt{\frac{L}{\mu}}\log\frac{1}{\epsilon})$.
- *Total number of edges activated to reach ϵ precision*: $\mathcal{O}(|\mathcal{E}|\sqrt{\rho}\sqrt{\frac{L}{\mu}}\log\frac{1}{\epsilon})$.

If $\kappa = \mathcal{O}(\frac{|\mathcal{E}|}{n})$, using Lemma 3.3, we see that, for any Laplacian $\mathcal{L}$:

$$\frac{1}{2}\sqrt{2\chi_1[\mathcal{L}]\chi_2[\mathcal{L}]}\mathrm{Tr}\,\mathcal{L} = \mathcal{O}(|\mathcal{E}|\sqrt{\rho})$$

Consequently, the communication complexity of DADAO run for $\mathcal{O}(\sqrt{\frac{L}{\mu}}\log\frac{1}{\epsilon})$ iterations is better than MSDA. Furthermore, for the complete graph, for any Laplacian $\mathcal{L}$ with $\rho[\mathcal{L}] = \mathcal{O}(1)$ (remind that, by definition $\rho \geq 1$), we have $\sqrt{2\chi_1[\mathcal{L}]\chi_2[\mathcal{L}]}\mathrm{Tr}\,\mathcal{L} = \mathcal{O}(n)$ whereas $|\mathcal{E}| = \mathcal{O}(n^2)$, thus DADAO uses an order of magnitude less communications than MSDA need to.

Comparison with the Continuized framework (Even et al., 2021a). In this framework and using their notations, we have:

- *Gossip matrices*: all Laplacian matrices $\mathcal{L}$ verifying $\mathrm{Tr}\mathcal{L} = 2$.
- *Total number of gradients to reach ϵ precision*: $\mathcal{O}(\frac{1}{\theta'_{\text{ARG}}}\sqrt{\frac{L}{\mu}}\log\frac{1}{\epsilon})$.
- *Total number of edges activated to reach ϵ precision*: $\mathcal{O}(\frac{1}{\theta'_{\text{ARG}}}\sqrt{\frac{L}{\mu}}\log\frac{1}{\epsilon})$.

First, we slightly rephrase one of the proposition of (Even et al., 2021a) to match our notations:

Lemma D.1. *For a Laplacian $\mathcal{L}$ with $\mathrm{Tr}\,\mathcal{L} = 2$, the communication and gradient complexity of the continuized framework is given by:*

$$\mathcal{O}(\sqrt{\chi_1[\mathcal{L}]\chi_2[\mathcal{L}]}\sqrt{\frac{L}{\mu}}\log\frac{1}{\epsilon})\,.$$

Proof. Under the notation of (Even et al., 2021a), $\theta'_{\text{ARG}} = \sqrt{\mu_{\text{gossip}}/\max_{\{v,w\}}\frac{R_{vw}}{P_{vw}}}$, with $\mu_{\text{gossip}} = \frac{1}{\chi_1[\mathcal{L}]}$ in our setting. Moreover, we remind that in (Even et al., 2021a), $\mathcal{L} = AA^\mathsf{T}$ and that $Ae_{vw} = \sqrt{P_{vw}}(e_v - e_w)$. Thus, by definition:

$$\begin{aligned}
\frac{R_{vw}}{P_{vw}} &\triangleq \frac{e_{vw}^\mathsf{T}A^+Ae_{vw}}{P_{vw}} && (69)\\
&= \frac{e_{vw}^\mathsf{T}A^+(e_v - e_w)}{\sqrt{P_{vw}}} && (70)\\
&= \frac{(A^{+\mathsf{T}}e_{vw})^\mathsf{T}(e_v - e_w)}{\sqrt{P_{vw}}} && (71)\\
&= \frac{((AA^\mathsf{T})^{+\mathsf{T}}Ae_{vw})^\mathsf{T}(e_v - e_w)}{\sqrt{P_{vw}}} && (72)\\
&= (e_v - e_w)^\mathsf{T}\mathcal{L}^+(e_v - e_w)\,, && (73)
\end{aligned}$$

and we have $\theta'_{\text{ARG}} = \frac{1}{\sqrt{2\chi_1[\mathcal{L}]\chi_2[\mathcal{L}]}}$. □

Next, if $\mathrm{Tr}\,\mathcal{L} = 2$, we proved in Lemma 3.3that $2\sqrt{\chi_1[\mathcal{L}]\chi_2[\mathcal{L}]} \geq (n-1)$. Thus, we see that the gradient complexity of DADAO in $\mathcal{O}(n\sqrt{\frac{L}{\mu}}\log\frac{1}{\epsilon})$ is always better than the one of the continuized framework. If we write $f(n) = \Omega(g(n))$ the fact that there is a constant $C > 0$ and $n_0 \in \mathbb{N}$ such that $\forall n \geq n_0,\ Cg(n) \leq f(n)$, then, in the cycle graph, for any $\mathcal{L}$ with $\mathrm{Tr}\,\mathcal{L} = 2$, $\chi_1[\mathcal{L}] = \Omega(n^3)$ and $\chi_2[\mathcal{L}] = \Omega(n)$. Thus DADAO uses an order of magnitude less gradients than the continuized framework for this graph.

□

E. Practical Implementation

In this section, we describe in more detail the implementation of our algorithm. As we did not physically execute our method on a compute network but carried it out on a single machine, all the asynchronous computations and communications had to be simulated. Thus, we will first discuss the method we followed to simulate our asynchronous framework before detailing the practical steps of our algorithm through a pseudo-code.

E.1. Simulating the Poisson Point Processes

To emulate the asynchronous setting, before running our algorithm, we generate 2 independent sequences of jump times at the graph's scale: one for the computations and one for the communications. As we considered independent P.P.Ps, the time increments follow a Poisson distribution. At the graph's scale, each node spiking at a rate of 1, the Poisson parameter for the gradient steps process is n. Following the experimental setting of the Continuized framework (Even et al., 2021a), we considered that all edges in $\mathcal{E}$ had the same probability of spiking. Thus, given any graph $\mathcal{E}$ and $\mathcal{L}$ its corresponding standard Laplacian with unit edge weights, we computed the parameter λ of the communication process:

$$\lambda = \sqrt{2\chi_1\left[\frac{\mathcal{L}}{|\mathcal{E}|}\right]\chi_2\left[\frac{\mathcal{L}}{|\mathcal{E}|}\right]}. \tag{74}$$

Having generated the 2 sequences of spiking times at the graph's scale, we run our algorithm playing the events in order of appearance, attributing the *location* of the events by sampling uniformly one node if the event is a gradient step or sampling uniformly an edge in $\mathcal{E}$ if it is a communication.

E.2. Pseudo Code

We keep the notations introduced in Eq. (3) and recall the following constant values specified in Appendix C.1:

$$\begin{array}{llllll}
\eta = \frac{1}{8}\sqrt{\frac{\nu}{L}} & \gamma = \frac{1}{4L} & \delta = \frac{1}{4}\sqrt{\frac{\nu}{L}} & \alpha = \frac{1}{4}\sqrt{\frac{\nu}{L}} & \beta = \frac{1}{2} & \theta = \frac{1}{2}\sqrt{\frac{L}{\nu}} \\
\tilde{\eta} = \frac{1}{8}\sqrt{\frac{\nu}{L}} & \tilde{\gamma} = \frac{1}{4\sqrt{\nu L}} & \tilde{\delta} = 1 & \tilde{\alpha} = \frac{1}{8}\sqrt{\frac{\nu}{L}} & \tilde{\beta} = 2\chi_1[\Lambda]\sqrt{\frac{L}{\nu}} & \nu = \frac{\mu}{2}
\end{array}$$

For the sake of completeness, we also specify the matrix $\mathcal{A}$ describing the linear ODE (8):

$$\mathcal{A} = \begin{pmatrix}
-\eta & \eta & 0 & 0 & 0 & 0 \\
\tilde{\eta} & -\tilde{\eta} & 0 & 0 & 0 & 0 \\
0 & 0 & -\alpha & \alpha & 0 & 0 \\
0 & -\theta\nu & -\theta & 0 & -\theta & 0 \\
0 & 0 & 0 & 0 & -\alpha & \alpha \\
0 & 0 & 0 & 0 & \tilde{\alpha} & -\tilde{\alpha}
\end{pmatrix} \tag{75}$$

As described in Appendix E.1, we call `PPPspikes` the process mentioned above, returning the ordered sequence of events and time of spikes of the two P.P.Ps. Then, we can write the pseudo-code of our implementation of the DADAO optimizer in Algorithm 2.

Algorithm 2: Pseudo-code of our implementation of DADAO on a single machine.

Input: On each machine $i \in \{1, ..., n\}$, an oracle able to evaluate ∇f_i, Parameters $\mu, L, \chi_1, t_{\max}, n, \lambda$.
The graph $\mathcal{E}$.

Initialize on each machine $i \in \{1, ..., n\}$:
  Set $X^{(i)} = (x_i, \tilde{x}_i, \tilde{y}_i)$ and $Y^{(i)} = (y_i, z_i, \tilde{z}_i)$ to 0 ;
  Set constants $\nu, \tilde{\eta}, \eta, \gamma, \alpha, \tilde{\alpha}, \theta, \delta, \tilde{\delta}, \beta, \tilde{\beta}$ using μ, L, χ_1;
  Set $\mathcal{A}$;
  $T^{(i)} \leftarrow 0$;
$\texttt{ListEvents}, \texttt{ListTimes} \leftarrow \texttt{PPPspikes}(n, \lambda, t_{\max})$;
$n_{\text{events}} \leftarrow |\texttt{ListEvents}|$;
for $k \in [\![1, n_{\text{events}}]\!]$ **do**
  if $\texttt{ListEvents}[k]$ *is to take a gradient step* **then**
    $i \sim \mathcal{U}([\![1, n]\!])$;
    $\begin{pmatrix} X^{(i)} \\ Y^{(i)} \end{pmatrix} \leftarrow \exp\left((\texttt{ListTimes}[k] - T^{(i)})\mathcal{A}\right) \begin{pmatrix} X^{(i)} \\ Y^{(i)} \end{pmatrix}$;
    $g_i \leftarrow (\nabla f_i(x_i) - \nu x_i - \tilde{y}_i)$; // Local gradient computation.
    $x_i \leftarrow x_i - \gamma g_i$;
    $\tilde{x}_i \leftarrow \tilde{x}_i - \tilde{\gamma} g_i$;
    $\tilde{y}_i \leftarrow \tilde{y}_i + (\delta + \tilde{\delta}) g_i$;
    $T^{(i)} \leftarrow \texttt{ListTimes}[k]$;
  else if $\texttt{ListEvents}[k]$ *is to take a communication step* **then**
    $(i, j) \sim \mathcal{U}(\mathcal{E})$;
    $\begin{pmatrix} X^{(i)} \\ Y^{(i)} \end{pmatrix} \leftarrow \exp\left((\texttt{ListTimes}[k] - T^{(i)})\mathcal{A}\right) \begin{pmatrix} X^{(i)} \\ Y^{(i)} \end{pmatrix}$;
    $\begin{pmatrix} X^{(j)} \\ Y^{(j)} \end{pmatrix} \leftarrow \exp\left((\texttt{ListTimes}[k] - T^{(j)})\mathcal{A}\right) \begin{pmatrix} X^{(j)} \\ Y^{(j)} \end{pmatrix}$;
    $m_{ij} \leftarrow (y_i + z_i - y_j - z_j)$; // Message exchanged.
    $z_i \leftarrow z_i - \beta m_{ij}$;
    $\tilde{z}_i \leftarrow \tilde{z}_i - \tilde{\beta} m_{ij}$;
    $z_j \leftarrow z_j + \beta m_{ij}$;
    $\tilde{z}_j \leftarrow \tilde{z}_j + \tilde{\beta} m_{ij}$;
    $T^{(i)} \leftarrow \texttt{ListTimes}[k]$;
    $T^{(j)} \leftarrow \texttt{ListTimes}[k]$;
return $(x_i)_{1 \leq i \leq n}$, *the estimate of* x^* *on each worker* i.