Accelerated methods for distributed optimization

Hadrien Hendrikx

PSL
UNIVERSITÉ PARIS

THÈSE DE DOCTORAT
DE L'UNIVERSITÉ PSL

Préparée à l'Ecole Normale Supérieure

Méthodes Accélérées pour l'Optimisation Distribuée

Soutenue par
Hadrien Hendrikx
Le 20 septembre 2021

Ecole doctorale n° ED386
Mathématiques Hadamard

Spécialité
Informatique

Composition du jury :

Antonin, CHAMBOLLE	Directeur de Recherche, Paris Dauphine	*Président*
Julien, HENDRICKX	Professeur, UC Louvain	*Rapporteur*
Martin, TAKAC	Professeur, Lehigh University	*Rapporteur*
Devavrat, SHAH	Professeur, MIT	*Examinateur*
Francis, BACH	Directeur de Recherche, INRIA	*Directeur de thèse*
Laurent, MASSOULIE	Directeur de Recherche, INRIA	*Directeur de thèse*

1794
ENS
ÉCOLE NORMALE
SUPÉRIEURE

RÉSUMÉ

Les modèles d'apprentissage modernes ont généralement besoin de volumes de données conséquents afin de réaliser de bonnes prédictions, et sont donc généralement entraînés de manière distribuée, c'est-à-dire en utilisant de nombreuses unités de calculs. Cette architecture distribuée peut venir de la taille des données, leur sensibilité, ou simplement pour entraîner les modèles plus rapidement.

Cependant, les modèles d'apprentissage sont souvent entraînés en utilisant des méthodes d'optimisation stochastiques intrinsèquement séquentielles, qui utilisent de nombreux gradients bruités mais faciles à calculer. De plus, de nombreux algorithmes réutilisent de l'information passée afin d'accélérer la convergence, ce qui nécessite un haut niveau de synchronie et de partage d'information.

Cette thèse présente un ensemble de résultats permettant d'étendre les avancées récentes en optimisation stochastique et accélérée dans le cadre décentralisé, c'est à dire sans coordination centrale mais via un ensemble de communications pair à pair.

MOTS CLÉS

Optimisation, Calcul distribué, Optimisation Stochastique, Méthodes Décentralisées

ABSTRACT

In order to make meaningful predictions, modern machine learning models require huge amounts of data, and are generally trained in a distributed way, *i.e.*, using many computing units. Indeed, the data is often too large or to sensitive to be gathered and stored at one place, and stacking computing units increases the computing power.

Yet, machine learning models are usually trained using stochastic optimization methods, that perform a sequence of steps which are noisy but relatively easy to compute. Besides, many algorithms reuse past information to speed up convergence, which requires a high level of synchrony between agents.

This thesis presents a set of results that extend the recent advances from stochastic and accelerated convex optimization to the decentralized setting, in which there is no central coordination but only pairwise communications.

KEYWORDS

Optimization, Distributed Algorithms, Stochastic Optimization, Decentralized Methods

Summary. In order to make meaningful predictions, modern machine learning models require huge amounts of data, and are generally trained in a distributed way, *i.e.*, using many computing units. Indeed, many datasets are so big that they cannot even fit into the memory of a single computer. The data may also be sensitive, and the users that generate it may not want to share it with other people. Finally, stacking computing units increases the overall computing power, and thus allows for faster training.

Yet, machine learning models are usually trained using stochastic first-order optimization methods, that use a sequence of gradient approximations which are noisy but relatively easy to compute. By doing so, they rely on the fact that it is generally better to perform many small approximate steps rather than a few precise steps. Because of this sequential nature, and the fact that each individual step is rather inexpensive, leveraging parallel computing capability is a key challenge in this case. Besides, many algorithms rely on *acceleration*, which is an optimization technique that reuses past information to perform extrapolation steps, in order to speed up convergence through the use of momentum. Yet, this use of past information requires a high level of synchrony and knowledge sharing between computing agents which can be difficult to achieve.

Indeed, in the centralized communication model, there is a server that aggregates the information sent by the many workers, and that can perform such global coordination (though generally at the price of some synchrony). In the decentralized communication model, each node has its own parameter, and communicates only with its neighbours in a given graph. In this case, global coordination is much more difficult to achieve, and existing algorithms are both slower and harder to analyze than in the centralized setting.

The first set of contributions of this thesis focuses on the decentralized setting. An accelerated algorithm based on dual coordinate descent is first developed for the randomized gossip model, in which communication edges activate uniformly at random, instead of synchronously during global communication rounds. In particular, this extends the results that were obtained using polynomial (Chebyshev) acceleration for synchronous algorithms. Then, this algorithm is extended to incorporate a key aspect of convex optimization: stochastic updates with variance reduction. We show the optimality of this finite-sum algorithm in many settings (synchronous in particular) by exhibiting a matching lower bound. This thus closes the gap between centralized and decentralized optimization, and shows that even though decentralized algorithms are harder to design, they achieve the same performances. The dual framework used leads to very appealing rates but relies on very strong oracles. We thus give a decentralized stochastic variance-reduced algorithm that relies on the much weaker primal oracles, although it is obtained through a dual approach.

Another contribution of this thesis is to study communication complexity in the centralized setting. The relative regularity framework is leveraged to combine acceleration with statistical preconditioning (using some local data at the server), in order to make the most progress with each update. We also show tight bounds on the relative condition number, and develop stochastic extensions.

The final contribution of this thesis focuses on the problem of preserving privacy while sharing information in a peer-to-peer fashion. We consider the rumor spreading problem in a complete graph, and show tight bounds on the differential privacy guarantees that can be obtained. We also precisely characterize the trade-off between privacy guarantees and diffusion speed.

Résumé. Les modèles d'apprentissage modernes ont généralement besoin de volumes de données conséquents afin de réaliser de bonnes prédictions, et sont donc généralement entraînés de manière distribuée, c'est-à-dire en utilisant de nombreuses unités de calculs. En effet, de nombreux jeux de données sont tellement gros qu'ils ne peuvent être stockés dans la mémoire d'un unique ordinateur. Dans d'autres cas, les données sont sensibles, et ceux qui les génèrent acceptent de les utiliser pour entraîner des modèles, mais ne souhaitent pas les envoyer telles quelles sur un serveur distant. Enfin, multiplier les unités de calcul permet d'augmenter la puissance du système, et donc d'entraîner les modèles plus rapidement.

Cependant, les modèles d'apprentissage sont souvent entraînés en utilisant des méthodes d'optimisation stochastique de premier ordre. Ces méthodes sont intrinsèquement séquentielles puisqu'elles utilisent de nombreux gradients bruités mais faciles à calculer. De plus, de nombreux algorithmes utilisent un mécanisme d'*accélération*, qui réutilise de l'information passée afin d'extrapoler le paramètre courant, et donc d'accélérer l'algorithme à travers une certaine forme d'inertie. Toutefois, ceci nécessite un haut niveau de synchronie entre les différents agents, qui peut être coûteux à réaliser.

En effet, dans un modèle de communication centralisé, un serveur agrège l'information envoyée par les noeuds du réseau, et peut s'occuper de cette coordination globale (au prix d'une synchronisation accrue). Dans un modèle décentralisé, chaque noeud a son propre paramètre, et ne communique qu'avec ses voisins dans un graphe donné. Dans ce cas, une coordination globale est beaucoup plus difficile à mettre en place, et les algorithmes existants sont à la fois plus lents et difficiles à analyser.

Le premier ensemble de contributions de cette thèse concerne le modèle décentralisé. Un algorithme accéléré basé sur la descente par coordonnées dans le dual est développé pour des communications pair-à-pair aléatoires (au lieu de communications globales de tous les agents avec tous leurs voisins). Ceci étend les résultat précédemment obtenus pour des algorithmes synchrones, qui utilisaient de l'accélération polynomiale (Chebyshev). Ensuite, cet algorithme est généralisé pour incorporer un aspect crucial de l'optimisation convexe: l'utilisation de gradients stochastiques avec réduction de variance. Nous montrons l'optimalité de cet algorithme pour sommes finies dans de nombreux cas particuliers (en particulier le cas synchrone) grâce à une borne inférieure sur la vitesse d'exécution de ces algorithmes. Le cadre dual permet de développer des méthodes très rapides, mais basées sur des oracles difficiles à calculer. Ainsi, nous développons enfin un algorithme décentralisé avec réduction de variance qui n'utilise que des gradients du primal, bien qu'il soit obtenu par une approche duale.

Un second axe de contributions de cette thèse concerne l'optimisation centralisée. La notion de régularité relative est utilisée pour combiner l'accélération avec le préconditionnement statistique (réalisé en utilisant des données locales du serveur), dans le but de tirer le meilleur parti de chaque pas de gradient. Des bornes précises sur le conditionnement relatif du problème, ainsi que des extensions stochastiques de cette méthode sont aussi développées.

La dernière contribution de cette thèse étudie le problème du partage anonyme d'information dans des protocoles pair-à-pair. Le problème de diffusion de rumeurs dans un graphe complet est étudié, et des bornes (inférieures et supérieures) sur les garanties de confidentialité différentielle sont données. L'existence d'un compromis entre vitesse de diffusion et garanties de confidentialité est mise en évidence, et un algorithme paramétré permettant de régler le niveau de confidentialité souhaité est donné.

Remerciements

Contents

Contributions and outline

Chapter 1: In this opening Chapter, we give a brief introduction to the key concepts of this thesis. In particular, we start by going over basics of convex optimization, and describe several first-order methods such as stochastic, accelerated, and Bregman gradient algorithms. Then, we present the two main distributed optimization settings: centralized and decentralized. Then, we focus on decentralized algorithms, and present the main approaches to obtain them. This manuscript is based on the papers that were published during this thesis. Thus, a significant effort has been dedicated to the writing of this chapter, which is intended to highlight the links between the other chapters, and how they consistently contribute to bringing the advances of convex optimization to the distributed (and in particular the decentralized) setting.

Chapter 2: A very simple decentralized optimization problem is decentralized averaging, in which all nodes start with a value, and we would like to compute the average of all the original values using pairwise interactions only. Interestingly, this problem already captures most of the difficulty of decentralized optimization. While accelerated methods based on Chebyshev polynomial were developed for the synchronous setting, it remained unknown whether the same accelerated rates could be obtained using randomized communications. We answer this question positively in this chapter, and show that this result actually extends to more general optimization problems as long as dual gradients can be computed. This Chapter is based on the paper *Accelerated Decentralized Optimization with Local Updates for Smooth and Strongly Convex Objectives* [Hendrikx, Bach, and Massoulié, 2019a], published at AISTATS 2019.

Chapter 3: In this chapter, we show that when the local objectives are finite sums, then a fine-grained dual formulation with an augmented graph intuition can be used to derive an algorithm that uses stochastic gradients locally. This requires more advanced optimization tools than Chapter 2 (accelerated stochastic block coordinate descent for composite objectives), as well as a finer understanding of synchronization times. Yet, we show that this leads to an optimal algorithm in many settings, and in particular under synchronous communications (which is a special case of stochastic communications in which the whole graph is sampled at once). This chapter is based on the paper *An Optimal Algorithm for Decentralized Finite Sum Optimization* [Hendrikx, Bach, and Massoulie, 2020b], accepted for publication at the Siam Journal on Optimization (SIOPT), which is itself based on the paper *An Accelerated Decentralized Stochastic Proximal Algorithm for Finite Sums* [Hendrikx et al., 2019b], published at NeurIPS 2019.

Chapter 4: This chapter closes the decentralized optimization contributions from this thesis, and focuses on obtaining a "dual-free" algorithm (that only requires primal oracles), from a dual approach. This is performed using Bregman coordinate descent with well-chosen reference functions, and showing that the dual problem is relatively smooth under this choice.

This Chapter is based on the paper *Dual-Free Stochastic Decentralized Optimization with Variance Reduction* [Hendrikx, Bach, and Massoulié, 2020a], published at NeurIPS 2020.

Chapter 5: This chapter is devoted to cutting communication complexity in the centralized setting. This is achieved by performing "statistical preconditioning": the server uses a (small) local dataset to precondition the updates, which is efficient if the local dataset approximates the global one. We show that the complexity of this algorithm depends on the relative condition number between the local objective of the server and the global objective, and that a square root dependence on this relative condition number can (asymptotically) be achieved by an accelerated algorithm. Finally, we precisely quantify this condition number when the dataset of the server is drawn i.i.d. from the same distribution as the global dataset. Although computationally more demanding, this method can cut the communication cost by a factor proportional to the size of the server's dataset. This Chapter is based on the paper *Statistically Preconditioned Accelerated Gradient Method for Distributed Optimization* [Hendrikx, Xiao, Bubeck, Bach, and Massoulie, 2020c], published at ICML 2020.

Chapter 6: In this chapter, we study the privacy guarantees of gossip protocols, in the special case of rumor spreading in complete graphs. We give matching lower and upper bounds on the differential privacy guarantees that can be achieved, and exhibit a trade-off between spreading speed and privacy. Finally, we give a parametrized protocol that allows to choose where to stand on this trade-off, and show that near-optimal privacy guarantees can be achieved while retaining logarithmic convergence speed. This Chapter is based on *Who started this rumor? Quantifying the natural differential privacy of gossip protocols* [Bellet, Guerraoui, and Hendrikx, 2020], published at DISC 2020.

Before going further, we would like to insist on the fact that code for all papers is available online.

CHAPTER 1

Introduction

Machine Learning is a field that aims at making predictions, based on data. Typical problems are for instance regression problems (determining the price of a house based on its characteristics, such as size, location, or number of rooms), or classification problems (determining whether a given picture represents a cat or a dog). To answer these questions, prediction models generally need several parameters to be tuned in order to be accurate, and assess the impact of a given variable on the final decision. For instance, how much does the price of a house increase when its size increases? These parameters can be set using expert knowledge, but it can be quite costly: in this case several real-estate agents are probably needed to help setting them. Machine learning algorithms learn these parameters directly from data. More specifically, given a model (*e.g.*, linear), and a dataset (all the houses that were sold on Craigslist or Leboncoin), the goal is to pick the model parameters that best explain the dataset (give the more accurate prices for the houses). Thus, expert knowledge is replaced by data when building the prediction model, and more data generally means better performances.

The process we have described has two major components. The first one consists in designing the model (what parameters should be learned, and what do they influence), and the evaluation rule (how do we measure the error between the predictions of our model and the ground truth). The study of this part is called *statistical modelling*, and it consists in designing the problem such that the quality of predictions increases rapidly with the size of the dataset. Then comes the *optimization* part: given a model and an error criterion, how do we find the parameters that yield the minimal error on a given dataset? Although there are interplays between the two (by trying to find models that are both easy to optimize and have good learning properties), this thesis only focuses on the second aspect, in the context of distributed systems.

Indeed, consider now that we would like to build a next word prediction model for cellphone keyboards, or an injury classification model based on radio images. In the first case, the dataset is made of all the words typed by all the cellphone users with the same language and keyboard application, together with information about the user. This is huge, and does not fit into the memory of a single computer. Yet, we would like to leverage all this data, and thus need to learn from it in a distributed way. In the second case, people may not want to send their medical data directly to the server, but would prefer to send a partial model that has been locally udpated using their data. Another reason one would use distributed computing is to increase the processing power: one can hope to achieve faster training by using more computing units that can make progress in parallel.

For all these reasons, distributed optimization is now a key component of machine learning systems, and comes with a variety of challenges. In this chapter, we present the main

tools and concepts that we rely on in this manuscript. We first start by describing convex optimization with a single machine, and then present the different distributed communication models, and how the algorithms that we have shown extend to these models.

1.1. Basics of Convex Optimization

We start this chapter by introducing the basic convex optimization concepts that we will use throughout the thesis. In order to present a self-contained introduction, all results are given with short proofs of increasing complexity. More detailed introductions can be found for instance in Boyd et al. [2004], Nesterov [2013c], Bubeck [2015]. We note $\mathcal{C}^p(\mathbb{R}^d)$ the set of p times continuously differentiable functions from $\mathbb{R}^d$ into $\mathbb{R}$, and study the following problem for $f \in \mathcal{C}^p(\mathbb{R}^d)$ with $p \geq 1$:

$$\min_{x \in \mathbb{R}^d} f(x). \tag{1.1.1}$$

Note that we do not consider non-smooth functions, and refer the interested reader to [Nesterov, 2013c, Chapter 3] in this case. We assume that Problem (1.1.1) is well-posed, *i.e.*, that $f_\star = \inf_{x \in \mathbb{R}^d} f(x)$ is finite, and that there exists $x_\star \in \mathbb{R}^d$ such that $f(x_\star) = f_\star$. We focus on *first-order* methods, meaning that we assume that we have access to ∇f, the gradient of f. We will sometimes assume stronger oracles, such as the gradient of the convex conjugate of f, or its proximal operator (that we will introduce later). A very simple solution to find the solution to Problem (1.1.1), consists in performing *gradient descent*. More precisely, we start from some $x_0 \in \mathbb{R}^d$, and perform for some sequence $\eta_t \geq 0$ the following algorithm:

$$x_{t+1} = x_t - \eta_t \nabla f(x_t). \tag{1.1.2}$$

Since the gradient is a descent direction, we know that if η_t is small enough then $f(x_{t+1}) \leq f(x_t)$, meaning that the objective function decreases. At this point, several questions arise:

(1) Is it possible to quantify "small enough", *i.e.*, how do we set η_t?
(2) Does the sequence $(x_t)_{t \geq 0}$ converge, and where to?
(3) Is it possible to certify that $f(x_t) \leq E_t$ for some $E_t \in \mathbb{R}$?

These questions cannot be answered non-trivially for general objective functions f. We thus make assumptions on the regularity of the objective function, that will allow us to analyze the gradient descent algorithm.

1.1.1. Regularity assumptions.

In order to guarantee that there exists non-trivial step-sizes such that gradient descent converges, one needs to make assumptions on the gradients. Indeed, we want to know how long the gradient information queried at point $x \in \mathbb{R}^d$ remains relevant, which is directly related to how fast the gradient changes. In particular, we assume that f is smooth, *i.e.*, that its gradient is Lipschitz. Note that unless explicitly stated, $\|\cdot\|$ refers to the Euclidean norm.

Definition 1 (Smoothness). *A function $f \in \mathcal{C}^1(\mathbb{R}^d)$ is said to be L-smooth if the following condition is satisfied for all $x, y \in \mathbb{R}^d$:*

$$\|\nabla f(x) - \nabla f(y)\| \leq L \|x - y\|. \tag{1.1.3}$$

This condition is equivalent [Nesterov, 2013c] to:

$$|f(x) - f(y) - \nabla f(y)^\top (x - y)| \leq \frac{L}{2} \|x - y\|^2. \tag{1.1.4}$$

We will see that the step-size sequence $(\eta_t)_{t\in\mathbb{N}}$ for which gradient descent converges is directly related with the smoothness constant L. We now introduce the notion of convexity.

DEFINITION 2. *A function $f \in \mathcal{C}^1(\mathbb{R}^d)$ is said to be convex if for any $x, y \in \mathbb{R}^d$:*

$$f(x) \geq f(y) + \nabla f(y)^\top (x - y). \tag{1.1.5}$$

It is said to be strictly convex if the inequality is strict for all $x \neq y$.

If $x \in \mathbb{R}^d$ is a local minimum, then convexity ensures that $f(y) \geq f(x)$ for $x \in \mathbb{R}^d$ since $\nabla f(x) = 0$, and so all local minima are actually global minima (such that $f(x) = f_\star$). This is a key property, as it prevents first-order methods from being stuck in suboptimal local minima (since they do not exist). The notion of convexity can be strengthened as follows.

DEFINITION 3 (Strong convexity). *A function $f \in \mathcal{C}^1(\mathbb{R}^d)$ is said to be σ-strongly-convex if the following condition is satisfied for all $x, y \in \mathbb{R}^d$:*

$$f(x) - f(y) - \nabla f(y)^\top (x - y) \geq \frac{\sigma}{2}\|x - y\|^2. \tag{1.1.6}$$

Note that if $f \in \mathcal{C}^2(\mathbb{R}^d)$, then strong convexity and smoothness are equivalent to:

$$\sigma I_d \preccurlyeq \nabla^2 f(x) \preccurlyeq L I_d, \tag{1.1.7}$$

where $I_d \in \mathbb{R}^{d\times d}$ is the identity matrix of size d and for two symmetric matrices $A, B \in \mathbb{R}^{d\times D}$, $A \preccurlyeq B$ if $B - A$ is positive semi-definite. Before continuing further, we introduce the *Bregman divergence* of f, which is defined for points $x, y \in \mathbb{R}^d$ by:

$$D_f(x, y) = f(x) - f(y) - \nabla f(y)^\top (x - y). \tag{1.1.8}$$

This divergence has many useful properties, and in particular the fact that if f is convex, $D_f(x, y) \geq 0$ and D_f is convex in its first argument (but not necessarily in the second one). Using this definition, smoothness and strong convexity can simply be written as:

$$\frac{\sigma}{2}\|x - y\|^2 \leq D_f(x, y) \leq \frac{L}{2}\|x - y\|^2. \tag{1.1.9}$$

Note that $D_{\frac{1}{2}\|\cdot\|_2^2}(x, y) = \frac{1}{2}\|x - y\|_2^2$, and so the left and right hand side of (1.1.9) can also be expressed in terms of Bregman divergences. We will see in Section 1.1.5 that this allows for useful generalizations of these regularity conditions. Smoothness and strong convexity have graphical interpretations: they mean that at any point x_0, the function f can be lower bounded and upper bounded by quadratic functions that go through $f(x_0)$, and their Hessians are defined respectively by σ and L times the identity matrix. An illustration of this can be found on Figure 1, Before continuing, we now quickly introduce the notion of proximal operator, which is defined for a convex function $f \in \mathcal{C}^1(\mathbb{R}^d)$ by:

$$\mathrm{prox}_f(x) = \arg\min_{u\in\mathbb{R}^d} f(u) + \frac{1}{2}\|u - x\|^2. \tag{1.1.10}$$

We also introduce the convex conjugate of f, which is defined as:

$$f^*(x) = \sup_{u\in\mathbb{R}^d} u^\top x - f(u). \tag{1.1.11}$$

Note that convex conjugation has many interesting property, and in particular that if $f \in \mathcal{C}^2(\mathbb{R}^d)$ and is strictly convex on $\mathbb{R}^d$, then f^* is well-defined and $\nabla f(\nabla f^*(x)) = x$ for $x \in \mathbb{R}^d$ and $\nabla f^*(x) = \arg\max_{u\in\mathbb{R}^d} u^\top x - f(u)$. There are also many links between convex

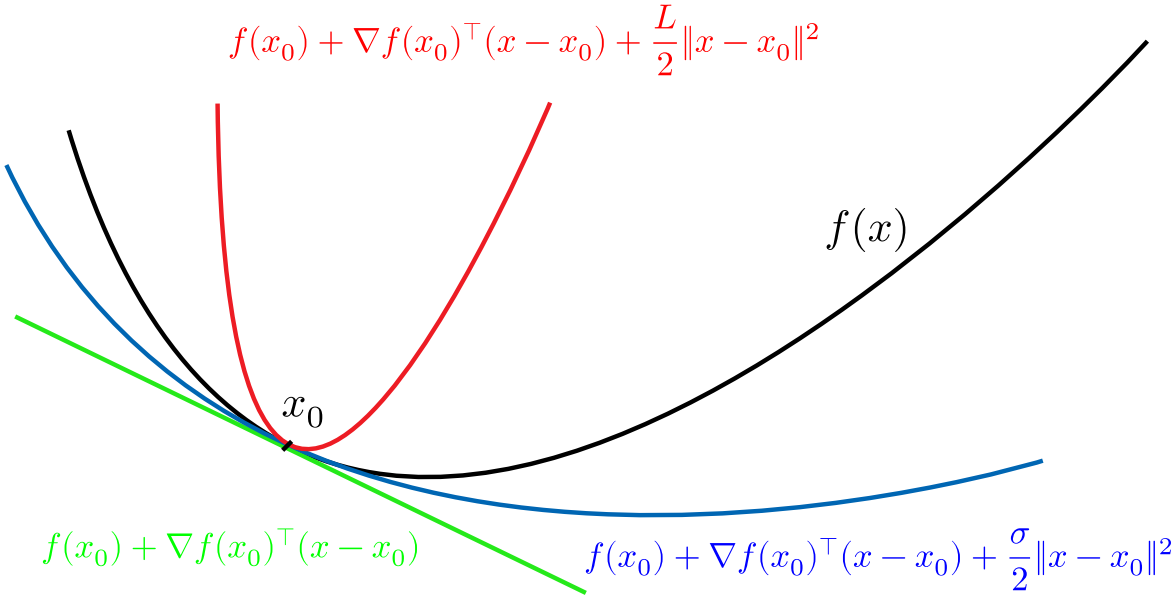


FIGURE 1. Graphical illustration of the quadratic upper-bound and lower bound provided respectively by L-smoothness (in red) and σ-strong-convexity (in blue) at point a x_0 for a function f.

conjugation and the proximal operator, and we refer the interested reader to Beck and Teboulle [2009b], Combettes and Pesquet [2011], Parikh and Boyd [2014], Chambolle and Pock [2016]. Note that the proximal operator and the convex conjugate can be defined for less regular functions, but we do not discuss them in this manuscript to avoid discussing well-posedness issues. To make things more concrete, we give an example of typical objective function that respects these assumptions. In particular, we consider a classification problem: given a sample $z_i \in \mathbb{R}^d$, we would like to determine what label $y_i \in \{-1, 1\}$ our algorithm should predict. To do so, we use a *logistic regression* model, which computes the probability that z belongs to class 1 as:

$$p_x(y_i = 1|z_i) = \left(1 + \exp(-z_i^\top x)\right)^{-1}, \tag{1.1.12}$$

where $x \in \mathbb{R}^d$ is the parameter that we would like to learn, through the following optimization problem:

$$x^\star = \min_{x \in \mathbb{R}^d} \left\{ f(x) \doteq \frac{1}{m} \sum_{i=1}^{m} \log\left(1 + \exp(-y_i z_i^\top x)\right) + \frac{\lambda}{2}\|x\|^2 \right\}, \tag{1.1.13}$$

where λ is a regularization parameter that prevents overfitting, *i.e.*, accounts for the fact that data used to train the model might be slightly different from the data on which it will be used. One can easily verify that the logistic regression objective defined in (1.1.13) is

strongly convex and smooth, with parameters

$$\sigma = \lambda, \qquad L = \lambda + \frac{1}{4m}\lambda_{\max}\left(\sum_{i=1}^{m} z_i z_i^\top\right), \tag{1.1.14}$$

where $\lambda_{\max}(M)$ is the largest eigenvalue of the matrix $M \in \mathbb{R}^{d\times d}$. Now that we have introduced the basic regularity assumptions and concepts, we can start proving convergence results for our algorithms.

1.1.2. Convergence of Gradient Descent. In this section, we analyze gradient descent, as given by Equation (1.1.2), and prove its speed of convergence. We first show that smoothness allows to choose a step-size that ensures that gradient descent actually leads to a decrease in function values.

LEMMA 1 (Descent Lemma). *If $f \in \mathcal{C}^1(\mathbb{R}^d)$ is L-smooth, then if $x_{t+1} = x_t - \eta_t \nabla f(x_t)$ for $\eta_t \leq \frac{1}{L}$, it holds that:*

$$\frac{\eta_t}{2}\|\nabla f(x_t)\|^2 \leq f(x_t) - f(x_{t+1}). \tag{1.1.15}$$

PROOF. Using Equation (1.1.4),

$$f(x_{t+1}) - f(x_t) - \nabla f(x_t)^\top (x_{t+1} - x_t) = f(x_{t+1}) - f(x_t) + \eta_t \|\nabla f(x_t)\|^2 \leq \frac{\eta_t^2 L}{2}\|\nabla f(x_t)\|^2.$$

The result is obtained by using that $\eta_t L \leq 1$ and rearranging the terms. □

In particular, for some $T \in \mathbb{N}$ and with a constant step-size $\eta \leq 1/L$, we have that the iterates produced by gradient descent verify

$$\sum_{t=0}^{T} \|\nabla f(x_t)\|^2 \leq \frac{2}{\eta}\left[f(x_0) - f(x_\star)\right], \tag{1.1.16}$$

so that $\nabla f(x_t) \to 0$ as $t \to \infty$. Thus, under the smoothness assumption, gradient descent guarantees that the norm of the gradient converges to 0, so that $(x_t)_{t\geq 0}$ converges to a local minimum (since $f(x_t) \leq f(x_{t+1})$, the extremum cannot be a maximum unless the function is flat). Since f is also convex, we thus know that $f(x_t) \to f_\star$ when $t \to \infty$. Yet, this is not entirely satisfactory, as we cannot bound $f(x_t) - f_\star$ for a given $t \in \mathbb{N}$ yet. In other words, we do not know how fast convergence is. This is adressed in the following theorem, which is standard and very similar to Nesterov [2013c, Theorem 2.1.14] or Bubeck [2015, Theorem 3.10].

THEOREM 1. *If $f \in \mathcal{C}^1(\mathbb{R}^d)$ is L-smooth and σ-strongly convex, then if we choose $\eta_t \leq 1/L$ for all t, the iterates (x_t) produced by gradient descent are such that for any $x_\star \in \arg\min_{x\in\mathbb{R}^d} f(x)$:*

$$\|x_T - x_\star\|^2 \leq \prod_{t=0}^{T}\left(1 - \eta_t\sigma\right)\|x_0 - x_\star\|^2. \tag{1.1.17}$$

PROOF. We decompose the first term as:

$$\|x_{t+1} - x_\star\|^2 = \|x_t - x_\star\|^2 - 2\eta_t \nabla f(x_t)(x_t - x_\star) + \eta_t^2\|\nabla f(x_t)\|^2.$$

Then, using the definition of the Bregman divergence of f, we write that:

$$-\nabla f(x_t)(x_t - x_\star) = f(x_\star) - f(x_t) - D_f(x_\star, x_t), \tag{1.1.18}$$

so that:

$$\|x_{t+1} - x_\star\|^2 \leq \|x_t - x_\star\|^2 - 2\eta_t D_f(x_\star, x_t) - 2\eta_t \left[f(x_t) - f(x_\star)\right] + \eta_t^2 \|\nabla f(x_t)\|^2 \tag{1.1.19}$$

Combining Lemma 1 with Equation (1.1.19), we obtain that:

$$\|x_{t+1} - x_\star\|^2 \leq \|x_t - x_\star\|^2 - \eta_t D_f(x_\star, x_t) - 2\eta_t \left[f(x_{t+1}) - f(x_\star)\right] \tag{1.1.20}$$

The result is obtained by using the strong convexity of f to bound the $D_f(x_\star, x_t)$ term, using the fact that $f(x_{t+1}) \geq f(x_\star)$, and chaining the inequality for all $t \leq T$. □

When $\sigma = 0$, the guarantees from Theorem 1 are trivial. Yet, the proof can be slightly adapted to recover similar guarantees.

THEOREM 2. *If $f \in \mathcal{C}^1(\mathbb{R}^d)$ is convex and L-smooth, and $\eta_t \leq 1/L$,*

$$f(x_T) - f(x_\star) \leq \frac{1}{2\sum_{t=0}^{T-1} \eta_t} \|x_0 - x_\star\|^2. \tag{1.1.21}$$

PROOF. Using that f is convex, Equation (1.1.20) can be rearranged as:

$$\eta_t \left[f(x_{t+1}) - f(x_\star)\right] \leq \frac{1}{2}\|x_t - x_\star\|^2 - \frac{1}{2}\|x_{t+1} - x_\star\|^2. \tag{1.1.22}$$

Summing this for all $t < T$, we obtain that:

$$\sum_{t=0}^{T-1} \eta_t \left[f(x_{t+1}) - f(x_\star)\right] \leq \frac{1}{2}\|x_0 - x_\star\|^2. \tag{1.1.23}$$

Then, the result is obtained by using that $f(x_T) \leq f(x_t)$ for all $t \leq T$, which is obtained from the descent lemma. □

We have seen that convergence rates can be obtained when $\sigma = 0$. Yet, to reach an error ε, one needs to perform $O(1/\varepsilon)$) iterations (sublinear convergence rate) instead of $O(\log \varepsilon^{-1})$ (linear convergence) when $\sigma > 0$. In the remainder of this thesis, we will focus on the $\sigma > 0$ case, although most results have analogs when $\sigma = 0$. In general, similarly to the case of gradient descent, the algorithm and the condition on the step-size are the same, or at least very similar. Yet, only sublinear convergence is achieved when $\sigma = 0$, although the complexity can be improved to $O(1/\sqrt{\varepsilon})$ iterations using *acceleration*, that we detail in Section 1.2.1 for the strongly convex case.

We see in Theorem 1 that if we choose $\eta_t = 1/L$ for all $t \in \mathbb{N}$ (which is the optimal choice in this case), then we obtain that the convergence rate depends on the the condition number κ_f, which is defined as:

$$\kappa_f = \frac{L}{\sigma}. \tag{1.1.24}$$

The condition number of the objective function is a key quantity in first-order (strongly) convex optimization, as it generally dictates how fast methods are similarly to Theorem 1. We will see in Theorem 6 that the pace of any gradient method is indeed limited by the condition number. We provide a graphical intuition for the influence of the condition number in Figure 2.

(a) Small κ_f (b) Big κ_f

FIGURE 2. Influence of the condition number on how easy a function is to optimize. The black lines represent the level sets of a function f and the red arrows the steps taken by gradient descent. A large condition number means that the function grows much faster in some directions than in others, and so the level-sets have an ellipsoidal shape. In the well-conditioned case (Figure 2(a)), the gradients roughly point towards the optimum $x^\star$ so large steps can be taken, and convergence is fast. In the ill-conditioned case (Figure 2(b)), gradients point far from the optimum, and so small steps must be taken to ensure that gradient descent gets closer to the optimum. Thus, the optimization process is very slow, because small steps are taken in a rather bad direction.

1.1.3. Stochastic Gradient Descent. In some cases, the gradient ∇f may be very expensive to compute, but stochastic estimates can be obtained in a cheap way. This is for instance the case in many machine learning applications, in which the objective function is the average of a *loss* function (that measures how well a given model fits the data) over all samples in a dataset. Computing a full gradient is expensive, since one needs to compute the gradient of the loss for all samples of the dataset. Yet, a stochastic estimate of the gradient can be obtained by drawing a few samples at random. This is the fundamental principle of Stochastic Gradient Descent (SGD), introduced by Robbins and Monro [1951], which writes:

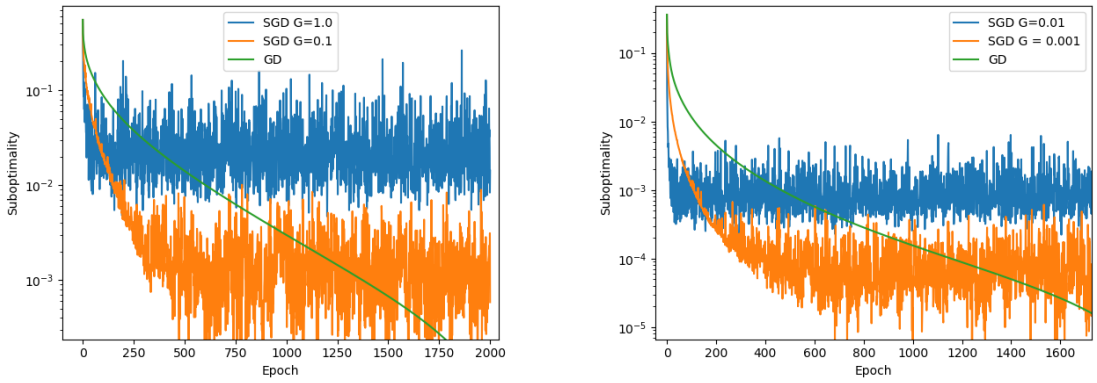
$$x_{t+1} = x_t - \eta_t g_t, \text{ with } \mathbb{E}[g_t] = \nabla f(x_t). \tag{1.1.25}$$

This method is widely used in practice, as it has very good performances for training large-scale models [Bottou, 2010]. It also benefits from very good statistical properties, as it is possible to avoid overfitting and obtain bounds on the testing error (the error on unseen data generated from the true distribution) if only one pass is performed (each training sample is used only once) [Nemirovski and Yudin, 1983, Polyak and Juditsky, 1992]. In particular, this avoids the need for explicit regularization, represented by the λ term in Equation (1.1.13). In order to analyze SGD, we need to assume further conditions on the stochastic estimates g_t. One of the simplest assumptions is that the variance of g_t is bounded.

ASSUMPTION 1. *The stochastic gradients g_t are independent and such that:*

$$\mathbb{E}[\|g_t - \nabla f(x_t)\|^2] \leq \chi^2 \tag{1.1.26}$$

This assumption is quite strong, as it requires the variance to be bounded at all times. This may be hard to achieve when f is strongly convex, since the magnitude of ∇f grows



(a) Small dataset ($N = 100$)

(b) Large dataset ($N = 10000$)

FIGURE 3. Comparison between deterministic and stochastic gradient descent on the LibSVM *a9a* dataset, for a regularization $\lambda = 10^{-3}$. SGD was run with step size $\eta = G/L$. We see that SGD decreases the error rapidly, but quickly saturates to mean error level that depends on G.

when x is far from $x_\star$. Under this assumption, the following convergence result holds [Robbins and Monro, 1951, Bottou et al., 2018].

THEOREM 3. *Under Assumption 1, the iterates produced by SGD with constant step-size* $\eta \leq \frac{1}{L}$ *guarantee:*

$$\mathbb{E}_t[\|x_T - x_\star\|^2] \leq (1 - \eta\sigma)^T \|x_0 - x_\star\|^2 + \frac{\eta\chi^2}{\sigma}. \tag{1.1.27}$$

PROOF. Using that $\mathbb{E}[g_t] = \nabla f(x_t)$, we obtain the following slightly perturbed version of Equation (1.1.19):

$$\mathbb{E}_t[\|x_{t+1} - x_\star\|^2] \leq \|x_t - x_\star\|^2 - 2\eta_t D_f(x_\star, x_t) - 2\eta_t \left[f(x_t) - f(x_\star)\right] + \eta_t^2 \mathbb{E}_t[\|g_t\|^2]. \tag{1.1.28}$$

Then, since $\mathbb{E}[g_t] = \nabla f(x_t)$:

$$\mathbb{E}[\|g_t\|^2] = \mathbb{E}[\|g_t - \nabla f(x_t)\|^2] + \|\nabla f(x_t)\|^2 \tag{1.1.29}$$

Thus, the exact same calculations as in the case of (deterministic) gradient descent can be performed, and we obtain:

$$\mathbb{E}_t[\|x_{t+1} - x_\star\|^2] \leq (1 - \eta_t\sigma)\|x_t - x_\star\|^2 + \eta_t^2\chi^2. \tag{1.1.30}$$

Taking a constant $\eta_t = \eta$, and using that $\sum_{t=0}^{T}(1-\eta\sigma)^k \leq 1/(\eta\sigma)$ leads to the final result. □

Thus, despite the fact that it only uses stochastic estimates of the gradient, SGD converges as fast as its deterministic counterpart, but only up to a given precision. At some point, the variance in the gradients takes over and SGD fails to optimize f beyond this error. We illustrate this phenomenon on Figure 3, in which we compare the GD (Equation (1.1.2)) and SGD (Equation (1.1.25)) algorithms on the Logistic Regression Problem of Equation (1.1.13). The *a9a* dataset was downloaded from the LibSVM repository[1].

[1] https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/binary.html

In order to obtain exact convergence to $x_\star$, one needs to employ other techniques, such as using a diminishing step-size, or averaging the iterates [Ruppert, 1988, Polyak and Juditsky, 1992]. Due to its huge practical impact, this method has been widely studied since [Bach and Moulines, 2011a, Dieuleveut et al., 2017, Gower et al., 2019]. Yet, one does not obtain the linear convergence rate that we show in Theorem 3 in this case, and only sublinear rates (in $O(T)$ at best without acceleration) can be achieved. This thesis focuses on convex problems, but SGD is also a widespread method both in theory and in practice for minimizing non-convex problems. In this case, convergence guarantees only hold for the norm of the gradient (in the form of Equation (1.1.16), plus a noise term).

1.1.4. Randomized Coordinate Descent. When the full gradient ∇f is expensive to compute, another way to obtain cheap estimates of the gradient can be to only perform coordinate updates (instead of full updates). This method is called randomized coordinate descent, and can be seen as a special case of stochastic gradient descent. In its most simple form, it writes, with $p_i \in \mathbb{R}$ the probability to pick coordinate i:

$$x_{t+1} = x_t - \frac{\eta_t}{p_i} \nabla_i f(x_t), \tag{1.1.31}$$

with $\nabla_i f(x_t) = e_i e_i^\top \nabla f(x_t)$, where $e_i \in \mathbb{R}^d$ is the unit vector corresponding to coordinate i. In the case of coordinate descent, the variance at optimum is $\|e_i^\top \nabla f(x_\star)\|^2 = 0$, and so one can mimick the deterministic gradient descent proof. In particular, we use the notion of *directional smoothness*, and say that f is L_i smooth in the direction i if for all $\delta > 0$,

$$D_f(x + \delta e_i, x) \leq \frac{L_i}{2} \delta^2. \tag{1.1.32}$$

Using this, the following theorem can be derived, where we denote $\mathbb{E}_t[\,\cdot\,] = \mathbb{E}[\,\cdot\,|x_t; I_t = i]$ to simplify notations, where I_t is the coordinate picked at time t.

THEOREM 4. *Denote* $L_t = \frac{1}{2}\mathbb{E}_t[\|x_t - x_\star\|^2 + \frac{\eta}{p_{\min}}[f(x_t) - f(x_\star)]]$, *with* $p_{\min} = \min_i p_i$. *Then, if* $\eta \leq p_i L_i$ *for all* i, *the iterates generated by coordinate descent (Equation* (1.1.31)*) verify:*

$$L_t \leq \max(1 - \eta\sigma, 1 - p_{\min})^t L_0. \tag{1.1.33}$$

PROOF. We start again from the main Equation (1.1.19):

$$\mathbb{E}_t[\|x_{t+1} - x_\star\|^2] \leq \|x_t - x_\star\|^2 - \eta D_f(x_\star, x_t) - 2\eta\,[f(x_t) - f(x_\star)] + \mathbb{E}_t[\frac{\eta^2}{p_i^2}\|\nabla_i f(x_t)\|^2] \tag{1.1.34}$$

This time, the update at time t has support on e_i, and so:

$$D_f(x_{t+1}, x_t) = f(x_{t+1}) - f(x_t) + \frac{\eta}{p_i}\|\nabla_i f(x_t)\|^2 \leq \frac{\eta^2 L_i}{2p_i^2}\|\nabla_i f(x_t)\|^2 \leq \frac{\eta}{2p_i}\|\nabla_i f(x_t)\|^2,$$

Thus, $f(x_{t+1}) \leq f(x_t)$, and so

$$\mathbb{E}_t[\frac{\eta^2}{p_i^2}\|\nabla_i f(x_t)\|^2] \leq \frac{2\eta}{p_{\min}}\left[f(x_t) - \mathbb{E}_t[f(x_{t+1})]\right]. \tag{1.1.35}$$

Plugging this into Equation (1.1.34), we obtain:

$$\frac{1}{2}\mathbb{E}_t[\|x_{t+1} - x_\star\|^2 + \frac{\eta}{p_{\min}}[f(x_{t+1}) - f(x_\star)]] \leq \frac{1 - \eta\sigma}{2}\mathbb{E}_t[\|x_t - x_\star\|^2] + \frac{\eta(1 - p_{\min})}{p_{\min}}\left[f(x_t) - f(x_\star)\right].$$

The result is obtained by taking the minimum of the two terms, and chaining the inequalities. □

Note that in this case, the Lyapunov function is not as simple as it was for gradient descent. Because of the stochastic aspect, function values terms are required to guarantee the decrease of the Lyapunov function L_t. Note that this introduces an additive $p_{\min}$ term in the complexity which was expected: without further assumptions, one cannot guarantee a given decrease of the objective if all coordinates have not been seen. Note that it is possible to choose the coordinates in a cyclic fashion (instead of uniformly at random), but the analysis is much more complex in this case [Saha and Tewari, 2013]. For more details on coordinate descent methods, and in particular for composite objectives, we refer the interested reader to Nesterov [2012], Richtárik and Takáč [2014], Lu and Xiao [2015]. Note that parallel and block versions can also be derived in a similar way [Richtárik and Takáč, 2016], but we discuss these methods in further details in Section 1.3.

1.1.5. Bregman gradients. To end this section, we present a last algorithm, which is known as Mirror Descent [Nemirovski and Yudin, 1983, Beck and Teboulle, 2003], or Bregman gradients. The iterations are the same, but the "Mirror Descent" terminology is usually used when the objective function is not smooth, whereas "Bregman gradients" is generally used together with the notion of *relative smoothness* (that we introduce in Definition 4). This algorithm is similar to gradient descent, but relies on a strictly convex potential function $h \in \mathcal{C}^2(\mathbb{R}^d)$ to perform its updates:

$$\nabla h(x_{t+1}) = \nabla h(x_t) - \eta_t \nabla f(x_t). \tag{1.1.36}$$

Another way to see these updates is to notice that the standard gradient descent iterations from (1.1.2) are equivalent to:

$$x_{t+1} = \arg\min_{x \in \mathbb{R}^d} \eta_t \nabla f(x_t)^\top x + \frac{1}{2}\|x - x_t\|^2. \tag{1.1.37}$$

In this form, we see that gradient descent consists in going in the direction of $-\nabla f(x_t)$, but without going too far from the initial point x_t in order to guarantee the descent condition. The "without going too far" part is measured in terms of Euclidean distance, but one could imagine measuring it differently, using Bregman divergences for instance, leading to the following iterations:

$$x_{t+1} = \arg\min_{x \in \mathbb{R}^d} \eta_t \nabla f(x_t)^\top x + D_h(x, x_t). \tag{1.1.38}$$

This is exactly the idea of Bregman gradient methods, and writing the optimality condition for the inner problem of (1.1.38) actually leads to (1.1.36), thus showing that these two iterations are indeed equivalent. Thus, Bregman gradients can be seen as standard gradient descent, on a space in which distances are measured using the function h, and is often referred to as a "non-Euclidean method". Mirror descent is widely used in the bandit and reinforcement learning communities, in which the parameters that are optimized represented probabilities, and for which the Kullback-Leibler divergence is very natural [Even-Dar et al., 2009, Bubeck, 2011, Hazan, 2012]. To take advantage of this non-Euclidean aspect, smoothness and strong convexity need to be adapted and defined relatively to the function h [Bauschke et al., 2017, Lu et al., 2018].

DEFINITION 4 (Relative smoothness and strong convexity). *A function f is $L_{f/h}$ smooth and $\sigma_{f/h}$ relatively strongly convex with respect to h if for all x, y,*

$$\sigma_{f/h} D_h(x, y) \leq D_f(x, y) \leq L_{f/h} D_h(x, y). \tag{1.1.39}$$

First note that when $h = \frac{1}{2}\|\cdot\|_2^2$, then the usual notions of smoothness and strong convexity are recovered (see Equation (1.1.9)). A very interesting consequence of this definition is that it allows to perform "smooth analyses" for non-smooth functions by changing the gradient step. For instance, some objectives (such as Poisson inverse problems) are non-smooth in the classical definition, but they are smooth with respect to the entropy. Thus, taking h as the entropy and performing Bregman gradients instead of gradient descent allows to recover a smooth behaviour. For instance, using relative smoothness, it is possible to prove the following lemma, which is a Bregman generalization of the cocoercivity inequality (a consequence of smoothness for convex functions).

LEMMA 2. *[Dragomir et al., 2021a] For all $\eta_t \leq \frac{1}{L_{f/h}}$, the following holds:*

$$D_{h^*}(\nabla h(x) - \eta_t \left[\nabla f(x) - \nabla f(y)\right], \nabla h(x)) \leq \eta_t D_f(y, x). \tag{1.1.40}$$

Using this Lemma, we can directly prove the following theorem, which can also be obtained directly [Bauschke et al., 2017, Lu et al., 2018].

THEOREM 5. *Let $T \in \mathbb{N}$. If $f \in \mathcal{C}^1(\mathbb{R}^d)$ is $L_{f/h}$ smooth and $\sigma_{f/h}$ strongly convex relative to h, and $\eta_t \leq L_{f/h}^{-1}$ for all $t \leq T$, then the iterates produced by T mirror descent iterations are such that:*

$$D_h(x_\star, x_T) \leq \prod_{t=0}^{T} (1 - \eta_t \sigma_{f/h}) D_h(x_\star, x_0). \tag{1.1.41}$$

The main difference in the proof is that in the Bregman framework, the natural notion of distance to the solution is given by $D_h(x_\star, x)$. Yet, $D_h(x_\star, x_{t+1})$ cannot be decomposed as easily as in the Euclidean case, but a Bregman analog of Equation (1.1.19) can still be obtained, as we show below.

PROOF. Define $V_t(x) = \eta_t \nabla f(x_t)^\top x + D_h(x, x_t)$, so that $\nabla V_t(x_{t+1}) = 0$. Then, we write that:

$$V_t(x_\star) - V_t(x_{t+1}) = D_{V_t}(x_\star, x_{t+1}) = D_h(x_\star, x_{t+1}), \tag{1.1.42}$$

using the fact that the Bregman divergences of V_t and h are equal (since they only differ by a linear term). The previous equation can be rearranged as:

$$D_h(x_\star, x_{t+1}) = D_h(x_\star, x_t) - D_h(x_{t+1}, x_t) + \eta_t \nabla f(x_t)^\top (x_\star - x_t) - \eta_t \nabla f(x_t)^\top (x_{t+1} - x_t). \tag{1.1.43}$$

Then, the last term can be expressed as:

$$-\eta_t \nabla f(x_t)^\top (x_{t+1} - x_t) = (\nabla h(x_{t+1}) - \nabla h(x_t))^\top (x_{t+1} - x_t) = D_h(x_{t+1}, x_t) + D_h(x_t, x_{t+1}),$$

and so we obtain a Bregman analog to Equation (1.1.19), which writes:

$$D_h(x_\star, x_{t+1}) = D_h(x_\star, x_t) - \eta_t \left[D_f(x_\star, x_t) + f(x_t) - f(x_\star)\right] + D_h(x_t, x_{t+1}). \tag{1.1.44}$$

Note that this equation is very similar to the base Equation (1.1.20), that we repeatedly used in the Euclidean case, and that it can also be obtained directly by using that $\nabla h(x_{t+1}) = \nabla h(x_t) - \eta_t \nabla f(x_t)$. Then, Lemma 2 tells us that:

$$D_h(x_t, x_{t+1}) \leq \eta_t \left[f(x_t) - f(x_\star)\right], \tag{1.1.45}$$

and so

$$D_h(x_\star, x_{t+1}) = D_h(x_\star, x_t) - \eta_t D_f(x_\star, x_t), \tag{1.1.46}$$

and we finish the proof using the relative strong convexity of f. □

Theorem 5 is very interesting in the sense that modifying the notions of distance and regularity from standard gradient descent does not change the convergence guarantees, but only the way suboptimality is measured. In particular, specifying Theorem 5 to the case $h = \frac{1}{2}\|\cdot\|^2$ exactly leads to Theorem 1. In Chapters 4 and 5, we show that the same thing happens when extending Bregman gradients to the stochastic setting (coordinate descent and stochastic gradient descent), so that the Euclidean convergence rates are recovered as specific cases. Now that we have presented a few basic first-order methods for convex optimization, we present more advanced techniques that improve the convergence guarantees.

1.2. Designing Faster methods

Gradient descent is a natural algorithm, and we have presented its convergence properties in the previous section. Yet, one may wonder whether gradient descent is optimal, or if it is possible to design algorithms that converge faster with the same assumptions.

1.2.1. Nesterov acceleration. In order to design such a method, a natural first question is the following: How fast can first-order methods be? Yet, to answer this, one must define the notion of speed. The common notion is that of *oracle complexity*. The complexity of an algorithm is measured by the number of calls to an oracle (which in our case is the gradient), to achieve a given precision level $\varepsilon > 0$. Then, one can design hard instances that cannot be optimized up to a given precision without a minimum number of calls to the oracle.

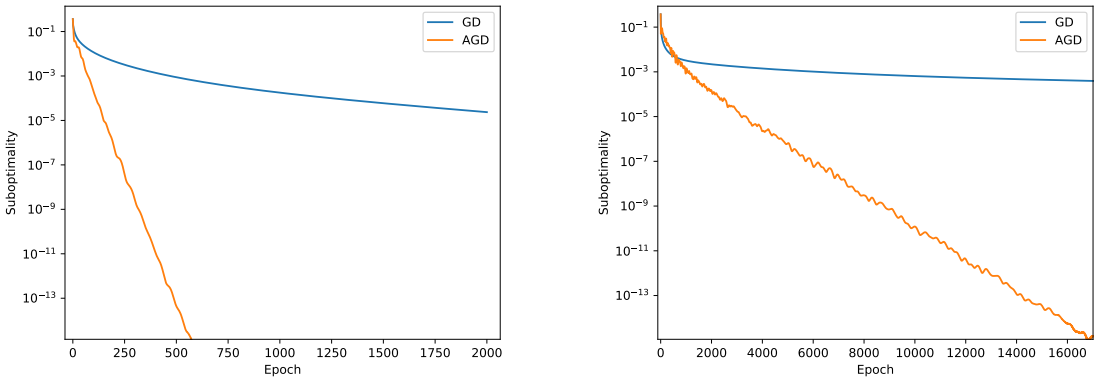
THEOREM 6. *[Nesterov, 2013c] For any $x_0 \in \mathbb{R}^\infty$ and any constants $\sigma > 0$, $\kappa_f > 1$ there exists a function $f \in \mathcal{C}^1(\mathbb{R}^\infty)$ which is σ-strongly convex and $\sigma\kappa_f$-smooth such that for any first-order method $\mathcal{M}$ satisfying $x_k \in x_0 + \mathrm{Lin}\{\nabla f(x_0), \cdots, \nabla f(x_{k-1})\}$ for all $k \geq 1$, we have (with $x_\star$ the minimum of f):*

$$\|x_k - x_\star\|^2 \geq \left(\frac{\sqrt{\kappa_f} - 1}{\sqrt{\kappa_f} + 1}\right)^{2k} \|x_0 - x_\star\|^2,$$

$$f(x_k) - f(x_\star) \geq \frac{\sigma}{2}\left(\frac{\sqrt{\kappa_f} - 1}{\sqrt{\kappa_f} + 1}\right)^{2k} \|x_0 - x_\star\|^2.$$

PROOF SKETCH. For $x \in \mathbb{R}^d$, we denote $nz(x)$ the highest index k such that $(x)_k \neq 0$. The proof proceeds by considering a specific function f such that $nz(\nabla f(x)) \leq nz(x) + 1$ and $(x_\star)_k = q^k$ for some $q > 0$. Then, starting from $x_0 = 0^d$, we know that after t calls to the oracle, and *regardless of the algorithm used*, it holds that $nz(x_t) \leq t$. In particular, $\|x_t - x_\star\|^2 \geq \sum_{k=t+1}^d \|(x_\star)_k\|^2 \approx q^{2t}$. The rest of the proof consists in picking f to achieve the desired constants. □

At this point, we know that gradient descent achieves the right order of convergence (iteration complexity that scales with $\log(1/\varepsilon)$), but there is a mismatch in the regularity constants. Gradient descent scales linearly with κ_f, whereas the lower bound scales as $\sqrt{\kappa_f}$. It could be that the lower bound is not tight, and that a better one can be achieved, but this



(a) High regularisation ($\lambda = 10^{-3}$)

(b) Small regularisation ($\lambda = 10^{-7}$)

FIGURE 4. Comparison between Gradient Descent (Equation 1.1.2) and Accelerated Gradient Descent (Equation (1.2.1)) for a regularized logistic regression problem with varying regularization. We see that AGD is drastically faster than GD, especially when λ is small. In particular, the GD curve is almost flat for $\lambda = 10^{-7}$, whereas AGD still converges quite fast.

is not the case as we will see. In particular, take $\beta > 0$ and Accelerated Gradient Descent (AGD), which is given by the following iterations:

$$\begin{aligned} x_{t+1} &= y_t - \frac{1}{L}\nabla f(y_t) \\ y_{t+1} &= x_{t+1} + \beta(x_{t+1} - x_t). \end{aligned} \tag{1.2.1}$$

This algorithm corresponds to the standard gradient descent algorithm, but in which an extrapolation step is added. It has the following convergence guarantees.

THEOREM 7. *[Nesterov, 2013c] If $\beta = \frac{\sqrt{\kappa_f}-1}{\sqrt{\kappa_f}+1}$, then the iterates produced by Equation (1.2.1) verify:*

$$f(x_k) - f(x_\star) \leq \left(1 - \sqrt{\frac{\sigma}{L}}\right)^k \left[f(x_0) - f(x_\star) + \frac{\sigma}{2}\|x_0 - x_\star\|^2\right]. \tag{1.2.2}$$

The proof of this theorem is out of scope of this section. The key point is that the convergence rate of this algorithm matches the lower bound from Theorem 6 (up to constants). This means that the lower bound is tight, and that Accelerated Gradient Descent (AGD) achieves the fastest possible rates in this class of algorithms. The complexity gap between gradient descent and accelerated gradient descent is significant, since the constant κ_f can be very large for standard machine learning problems. Thus, having a $\sqrt{\kappa_f}$ bound instead of κ_f one can make a problem tractable, or allow to use much smaller regularization for a given statistical learning problem (since κ_f scales as the inverse of the regularization). We illustrate this speedup numerically in Figure 4.

We have presented acceleration for the standard gradient descent method in the smooth and strongly convex setting, but it was originally introduced in the convex setting ($\sigma = 0$), in which case the iteration complexity changes from $O(1/\varepsilon)$ to $O(1/\sqrt{\varepsilon})$ [Nesterov, 1983].

Another famous accelerated method is the so-called *Heavy Ball Acceleration* [Polyak, 1964], but it enjoys weaker convergence guarantees in general. Nesterov acceleration can also be applied to many other settings, for instance to composite objectives, which incorporate a non-smooth term (but for which we know the proximal operator) [Beck and Teboulle, 2009b, Nesterov, 2013a, Chambolle and Dossal, 2015]. Similar ideas can also be applied to the proximal point algorithm (in which one iterates the proximal operator of the function instead of applying gradient descent) [Güler, 1992]. Note that a generic acceleration framework named *Catalyst* has also been developed, that allows to obtain comparable improvements in the convergence rate (up to logarithmic factors) with any base algorithm [Lin et al., 2015a]. This framework is based on an accelerated proximal point algorithm in which inner problems are solved by the algorithm that one wishes to accelerate. We refer the interested reader to d'Aspremont et al. [2021] for more details on accelerated methods.

We see that acceleration significantly speeds up first-order algorithms for ill-conditioned problems. A large part of this thesis is devoted to developping accelerated methods that are relevant for distributed optimization.

Accelerated Stochastic methods. Just like standard gradient descent, accelerated gradient descent can be used in stochastic settings. In this case, the following more general form of accelerated gradient descent is generally prefered, with g_t an approximation of the gradient:

$$\begin{aligned} y_t &= \frac{(1-\alpha_t)x_t + \alpha_t(1-\beta_t)v_t}{1-\alpha_t\beta_t} \\ v_{t+1} &= (1-\beta_t)v_t + \beta_t y_t - \frac{a_{t+1}}{B_{t+1}} g_t \\ x_{t+1} &= y_t + \alpha_t(v_{t+1} - (1-\beta_t)v_t - \beta_t y_t). \end{aligned} \tag{1.2.3}$$

This algorithm is robust to stochastic noise as long as the g_t are unbiased, *i.e.*, as long as $\mathbb{E}[g_t] = \nabla f(y_t)$. In this case, convergence can be proven for accelerated coordinate descent [Lin et al., 2015b, Nesterov and Stich, 2017, Allen-Zhu et al., 2016], which corresponds to the special case:

$$g_t = \frac{1}{p_i} e_i e_i^\top \nabla f(y_t). \tag{1.2.4}$$

In Chapter 3, we study variants of this algorithm that support strong convexity in semi-norms induced by non-PSD quadratics, together with proximal operators of a non-smooth term and arbitrary sampling (non-uniform choice of p_i). Other types of gradient estimates can be used, and in particular Vaswani et al. [2019] prove convergence of iterations (1.2.3) under a strong growth condition of the form

$$\mathbb{E}[\|g_t\|^2] \leq \rho \|\nabla f(x_t)\|^2, \tag{1.2.5}$$

for some constant $\rho > 0$.

Bregman gradients. We have seen that the lower bound for smooth and strongly convex optimization does not match the convergence rate of gradient descent, and that accelerated versions can be developed in this setting. The lower bound from Theorem 6 is still valid since the Bregman assumptions generalize that of gradient descent in the sense that it can be recovered by choosing $h = \frac{1}{2}\|\cdot\|^2$. Yet, the acceleration proofs are specific to gradient descent.

Recall that the convergence results from gradient descent could directly be generalized to the Bregman setting, so one may hope to obtain a complexity that is similar to the euclidean case. Yet, surprisingly, the opposite happens in this case: the lower bound from Theorem 6 is actually not tight in the case of arbitrary h, and can be strengthened as shown by Dragomir et al. [2021b]. In particular, one cannot hope to beat the simple Bregman gradient scheme presented in (1.1.36) without assuming more structure or regularity on the functions than just relative smoothness.

Yet, weaker forms of acceleration can be obtained under a favorable triangle scaling inequality [Hanzely et al., 2021], or when assuming that h is regular enough, as we detail in Chapter 5.

1.2.2. Variance reduction. We have seen in Section 1.1.3 that SGD only converges to a region around $x_\star$ because of the variance in the gradients. We focus in this section on the specific *finite sum* problem, in which the objective function f can be decomposed as:

$$f(x) = \frac{1}{m}\sum_{i=1}^{m} f_i(x), \tag{1.2.6}$$

where each function $f_i \in \mathcal{C}^1(\mathbb{R}^d)$ is L-smooth. This corresponds to the typical Empirical Risk Minimization (ERM) problem in which we have a model $x \in \mathbb{R}^d$, a dataset of samples (z_i) for $i \in \{1, \dots, m\}$, and a loss function ℓ, where $\ell(x, z)$ measures the error made by a model x when predicting a given sample z. In this case, the regularized ERM objective writes for $\lambda > 0$:

$$\frac{1}{m}\sum_{i=1}^{m} \ell(x, z_i) + \frac{\lambda}{2}\|x\|^2, \tag{1.2.7}$$

and the goal is to minimize it in x in order to obtain the model that achieves the lowest error on the training dataset. The regularization parameter $\lambda > 0$ allows to limit the model complexity and prevent overfitting, *i.e.*, having a model that performs well on the training data, but would perform poorly on new test data. An instanciation of SGD for this problem would be:

$$x_{t+1} = x_t - \eta_t \nabla f_{i_t}(x_t), \tag{1.2.8}$$

where i_t is a random index drawn at time t and $f_{i_t}(x) = \ell(x, z_{i_t}) + \frac{\lambda}{2}\|x\|^2$. Yet, the noise has a lot of structure in this case, since we know that it comes from taking the gradient of a specific function f_{i_t} instead of f, and we know which f_{i_t} it comes from.

Direct approaches. A direct approach for variance reduction is to perform the following updates:

$$x_{t+1} = x_t - \eta_t g_t, \tag{1.2.9}$$

where g_t is a variance-reduced gradient. Informally, this means that we would like to have that $g_t \to 0$ when $x_t \to x_\star$. Many such g_t are possible, and we detail here the main ones.

Stochastic Average Gradient (SAG) [Schmidt et al., 2017]. A very natural estimate g_t consists in taking

$$g_t = \frac{1}{m}\sum_{i=1}^{m} \nabla f_i(\phi_t^{(i)}), \tag{1.2.10}$$

where the $\phi_t^{(i)}$ are such that $\phi_t^{(0)} = x_0$, and then $\phi_{t+1}^{(i_t)} = x_t$, and $\phi_{t+1}^{(j)} = \phi_t^{(j)}$ for $j \neq i_t$. In other words, the algorithm stores the most recent $\nabla f_i(x_t)$ that it has computed for each i, and aggregates them to form the pseudo-gradient g_t. Ideally, one would like to have

$\phi_t^{(i)} = x_t$ for all i, but this would require recomputing the full gradient at each step, which we specifically would like to avoid. As x_t approaches the optimum, each $\phi_t^{(i)}$ approaches $x_\star$ and so $g_t \approx \frac{1}{n}\sum_{i=1}^m \nabla f_i(x_\star) = 0$. This algorithm corresponds to SAG, and although it has a very simple form, it is quite hard to analyze. This mainly comes from the fact that the updates are *biased*, in the sense that $\mathbb{E}[g_t] \neq \nabla f(x_t)$.

SAGA [Defazio et al., 2014a]. As previously stated, SAG provides a biased estimator of the gradient, and the method is thus very hard to analyze. Instead, SAGA uses a slightly different update:

$$g_t = \nabla f_i(x_t) - \nabla f_i(\phi_t^{(i)}) + \frac{1}{m}\sum_{i=1}^m \nabla f_i(\phi_t^{(i)}) \tag{1.2.11}$$

Similarly to SAG, $g_t \to 0$ as x_t converges to $x_\star$, but this time $\mathbb{E}_t[g_t] = \nabla f(x_t)$, so that the stochastic updates are unbiased. The analysis is thus similar to that of SGD, with an additional error term that vanishes this time.

Stochastic Variance Reduced Gradient (SVRG) [Johnson and Zhang, 2013b]. SAG and SAGA store the stochastic gradients corresponding to each component and use them to construct an estimate of the full gradient. Instead, the SVRG method maintains a running point $\tilde{\phi}_t$, which is periodically updated, and performs the following update:

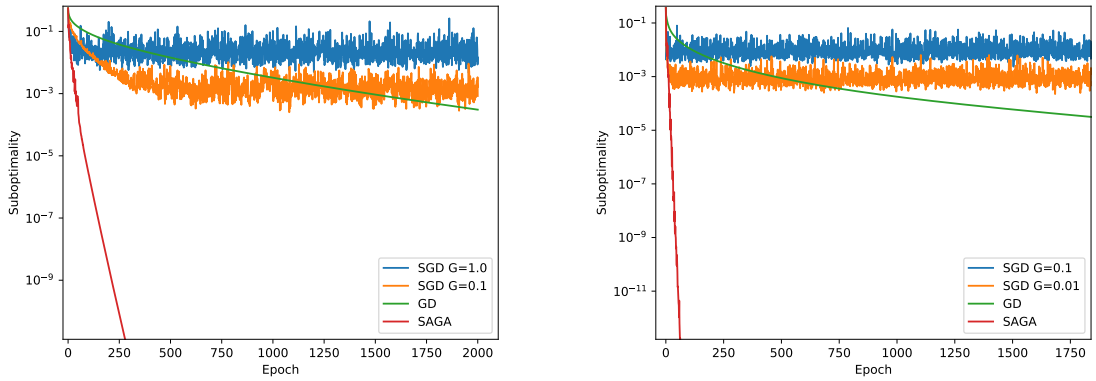
$$g_t = \nabla f_i(x_t) - \nabla f_i(\tilde{\phi}_t) + \nabla f(\tilde{\phi}_t). \tag{1.2.12}$$

If $\tilde{\phi}_t$ is updated every $K = O(m)$ rounds, then the overhead of computing a full gradient every K iterations is only of a constant factor, and the memory cost is significantly lower than SAG or SAGA since only $\tilde{\phi}_t$ and $\nabla f(\tilde{\phi}_t)$ need to be stored. Yet, this algorithm has a double-loop structure which makes it harder to tune in practice.

Other direct methods such as Finito [Defazio et al., 2014b] or MISO [Mairal, 2013] have also been introduced, but we do not detail them in this thesis. One key feature is that all these methods have very similar convergence rates. We refer the interested reader to the corresponding paper for each method, but a general convergence theorem can be stated informally as follows.

THEOREM 8 (Informal). *With an appropriate choice of parameters, the previous methods take time $T_\varepsilon = O((\kappa_s + m)\log \varepsilon^{-1})$ iterations to reach error ε, with $\kappa_s = L/\sigma$.*

Thus, stochastic variance-reduced methods enjoy the same iteration complexity as deterministic methods, but using much cheaper gradients. Note that there is a subtle point here, in the fact that κ_s is not the same condition number as previsouly introduced. Indeed, the "batch" condition number $\kappa = L_f/\sigma$ is defined using the condition number of f. Here, the condition number number κ_s is defined using the smoothness of each individual f_i. In particular, the smoothness of the average is different from the average smoothness, and so we have that $\kappa_b \leq \kappa_s \leq m\kappa_b$. In the worst case, $\kappa_s = m\kappa_b$, and so there is actually no speedup over deterministic methods, since computing one full gradient takes the same time as computing m stochastic gradients. Yet, stochastic variance-reduced methods converge much faster than gradient descent in practice, indicating that $\kappa_s << m\kappa_b$ in most practical settings. We explain this difference more in details in Chapter 3, and for now only illustrate this numerically on Figure 5, which uses the same setting as Figure 3 but now adds a comparison with the SAGA algorithm. We see that SAGA is as fast as SGD (when used with



(a) Small dataset ($N = 100$)

(b) Large dataset ($N = 10000$)

FIGURE 5. Comparison between deterministic, stochastic, and variance-reduced methods on the LibSVM *a9a* dataset, for a regularization $\lambda = 10^{-5}$.

the same step-size), but converges to the true optimum and not to a region determined by the variance.

Dual approach. This section describes the dual approach to variance reduction that leads to the Stochastic Dual Coordinate Ascent (SDCA) algorithm [Shalev-Shwartz and Zhang, 2013]. In particular, we study a variant of Problem (1.2.6) in which the regularization is explicit, and rewrite it equivalently as:

$$\begin{aligned} &\min_{x\in\mathbb{R}^{m\times d}} \frac{1}{m}\sum_{i=1}^{m} f_i(x_i) + \frac{\sigma}{2}\|x_0\|^2, \\ &\text{such that } x_0 = x_i \ \forall i \in \{1,\cdots,m\}. \end{aligned} \tag{1.2.13}$$

The equality constraints can be written $A^\top x = 0$ for some matrix $A \in \mathbb{R}^{m\times m}$ which is such that $(A^\top x)_i = \mu_i(x_0 - x_i)$ for some $\mu_i \in \mathbb{R}$. Thus, one can write the equality constraint using Lagrangian multipliers $y \in \mathbb{R}^{m\times d}$ in the following way:

$$\min_{x\in\mathbb{R}^{m\times d}} \sup_{y\in\mathbb{R}^{m\times d}} \frac{1}{m}\sum_{i=1}^{m} f_i(x_i) + \frac{\sigma}{2}\|x_0\|^2 - \frac{1}{m}\mathrm{Tr}\left(y^\top A^\top x\right). \tag{1.2.14}$$

Indeed, Problems (1.2.13) and (1.2.14) are equivalent since the supremum is equal to infinity if x is such that $A^\top x \neq 0$. From here, one can invert the min and the max and obtain:

$$\sup_{y\in\mathbb{R}^{m\times d}} -\frac{1}{m}\sum_{i=1}^{m} \max_{x_i\in\mathbb{R}^d} \left[x_i^\top (Ay)_i - f_i(x_i)\right] - \max_{x_0\in\mathbb{R}^d} \frac{1}{m} x_0^\top (Ay)_0 - \frac{\sigma}{2}\|x_0\|^2. \tag{1.2.15}$$

take the dual form of Problem (1.2.13), which writes:

$$\sup_{y\in\mathbb{R}^d} -\frac{1}{m}\left[\sum_{i=1}^{m} f_i^*((Ay)_i) + \frac{1}{2\sigma m}\|(Ay)_0\|^2\right], \tag{1.2.16}$$

where f_i^* is the convex conjugate of f_i, defined in Equation (1.1.11). Then, one can apply a coordinate descent method on the dual problem, and obtain an algorithm in which only one ∇f_i^* needs to be computed at each iteration, since $(Ay)_i = -\mu_i y_i$. This is a coordinate

descent algorithm, and so there is no residual variance $\chi > 0$, and one can show that the same rates as Theorem 8 are obtained with this algorithm.

There are two caveats to this method: the first one is that strong convexity needs to be explicit (and we need to know it to set the parameters of the algorithm), and the second and main one is that the oracle is ∇f_i^* instead of ∇f_i. Although ∇f_i^* might be easier to compute for some loss functions, it is in general much harder to evaluate, since it is as hard as minimizing function f_i itself (as can be seen from the characterization given after Equation (1.1.11)). Yet, it is possible to fix these issues and there are alternative versions of SDCA that can directly be formulated in a primal way [Shalev-Shwartz, 2016, He et al., 2018]. A large part of this thesis builds on a related dual approach, and Chapter 4 presents a decentralized method that is very similar to dual-free SDCA when used on a single machine.

1.2.3. Variance reduction and acceleration. When the objective function has the form of Equation (1.2.6), and a stochastic oracle is assumed (at each step $t \geq 0$, ∇f_{i_t} is computed, with i_t drawn at random), Lan and Zhou [2017] show a lower bound indicating that the number of iterations T_ε required to obtain an error smaller than a threshold $\varepsilon > 0$ is of order:

$$T_\varepsilon = O((m + \sqrt{m\kappa_s}) \log(\varepsilon^{-1})), \tag{1.2.17}$$

which is faster than the convergence rate discussed in Theorem 8. Yet, it is possible to combine acceleration and variance reduction to obtain an optimal finite-sum algorithm, as shown by Lan and Zhou [2017] in a primal-dual approach. This acceleration can also be obtained from a completely dual approach [Shalev-Shwartz and Zhang, 2014, Lin et al., 2014] by using accelerated (proximal) coordinate descent methods. Then, accelerated variants of direct (primal) variance-reduced methods were developped, and in particular Katyusha [Allen-Zhu, 2017], which is an acceleration of the SVRG method, and a direct acceleration of the SAGA method [Zhou, 2019]. In this thesis, we bridge the gap between convex and decentralized optimization by showing that these approaches (and in particular the dual ones) can be applied in the more general decentralized optimization setting. In particular, Chapter 3 proposes a decentralized method that reduces to APCG [Lin et al., 2014] when applied to a single-machine problem.

1.3. Distributed Optimization Models

In the previous section, we have presented a brief overview of first-order convex optimization algorithms. The common pattern was that they are all *incremental* algorithms, in which an oracle (generally the gradient of the objective function) is queried, and the information returned is used to construct a new estimate of the solution and a new point at which the gradient will be queried (which are often the same points). In the distributed model, this purely sequential nature changes, since operations can be performed in parallel on different machines. Thus, the algorithms that we consider heavily depend on the modelling assumptions that we make, and that depend on the following questions: where is the data? What communications are allowed? Is it possible to enforce synchrony? In this section, we present two main communication models: centralized and decentralized, and then discuss how some assumptions, such as synchronous communications, can be relaxed. Throughout this section, we consider that the system is made of n computing nodes, that node i holds function f_i,

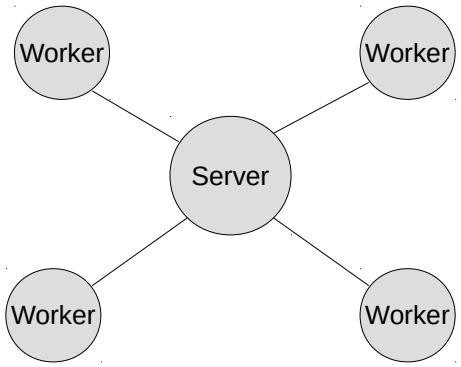


FIGURE 6. Centralized distributed system

and that the global objective is:

$$\min_{x\in\mathbb{R}^d} \frac{1}{n}\sum_{i=1}^{n} f_i(x). \tag{1.3.1}$$

1.3.1. Centralized model. A very simple communication model is the centralized one. In this model, there is a central server that can aggregate information from all the nodes, compute updates, and send the information back, as shown in Figure 1.3.1. In particular, thanks to the linearity of the gradients, $\nabla f(x) = \frac{1}{n}\sum_{i=1}^{n} \nabla f_i(x)$, so that one step of gradient descent in the centralized setting can be simply expressed like this:

(1) The server sends the current parameter x_t to the workers.
(2) Each worker i computes $g_t(i) = \nabla f_i(x_t)$, and sends it to the server.
(3) The server computes the new parameter $x_{t+1} = x_t - \frac{\eta_t}{n}\sum_{i=1}^{n} g_t(i)$.

Assuming each communication takes time τ, and each local gradient computation takes time 1, each iteration takes time $1+\tau$, and so the time $T^{\mathrm{GD}}_{\mathrm{centralized}}(\varepsilon)$ to reach precision ε is given by:

$$T^{\mathrm{GD}}_{\mathrm{centralized}}(\varepsilon) = O((1+\tau)\kappa\log(\varepsilon^{-1})). \tag{1.3.2}$$

If one uses accelerated gradient descent instead, then the results directly translate to this setting, and the new time complexity is:

$$T^{\mathrm{AGD}}_{\mathrm{centralized}}(\varepsilon) = O((1+\tau)\sqrt{\kappa}\log(\varepsilon^{-1})). \tag{1.3.3}$$

We see that in this simple centralized setting with full gradient algorithms, the results from single-machine convex optimization directly transfer to the centralized communication model. In particular, the single-machine time to convergence is given by:

$$T^{\mathrm{AGD}}_{\mathrm{serial}}(\varepsilon) = O(n\sqrt{\kappa}\log(\varepsilon^{-1})). \tag{1.3.4}$$

Thus, as long as the communication time is small enough (*i.e.*, $\tau < 1$), we have that:

$$T^{\mathrm{AGD}}_{\mathrm{centralized}}(\varepsilon) = O\left(\frac{T^{\mathrm{AGD}}_{\mathrm{serial}}(\varepsilon)}{n}\right), \tag{1.3.5}$$

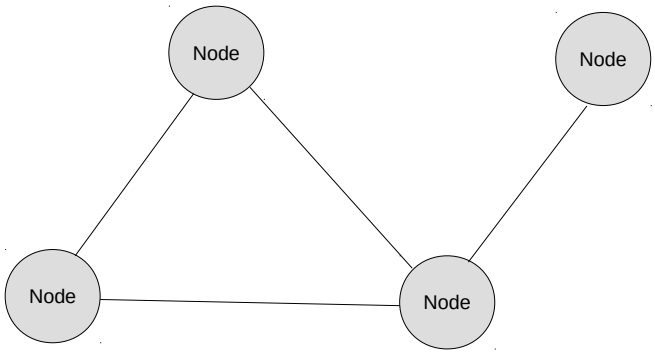


FIGURE 7. Decentralized distributed system

and so the accelerated gradient method achieves *linear speedup*, meaning that the algorithm is n times faster when run on n machines. One can also use a similar approach to leverage parallel computing power with stochastic gradient descent, in which case averaging reduces the variance of the gradient estimator by a factor n. The same methodology can also be applied to randomized block coordinate descent [Nesterov, 2012, Richtárik and Takáč, 2016, Fercoq and Richtárik, 2015, Hannah et al., 2018] by having each worker compute random coordinate gradients of the same function, and obtain a similar linear speedup. Yet, centralized algorithms are considered to be less robust because they have a single point of failure: if the server breaks down, then the whole algorithm stops. Besides, centralized systems cannot scale indefinitely: the server has a limited bandwidth and can only handle a limited number of workers at the same time. Past this point, workers need to wait, which increases the communication time. One way to deal with this communication bottleneck is to use asynchronous communications, that allow nodes to still perform useful computations while they are waiting for the response of the server, and which we present more in details in Section 1.3.4. Another orthogonal line of work consists in performing iterations that are more computationally intensive, but also much more efficient so that the overall communication complexity is lowered. These methods include CoCoA [Smith et al., 2018], and the DANE algorithm [Shamir et al., 2014, Yuan and Li, 2020], and are studied in more details in Chapter 5, in which we develop an accelerated method of this type with strong theoretical guarantees.

1.3.2. Decentralized model. The decentralized model relies on a completely different approach. First of all, nodes are assumed to be linked by a connected undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ where $\mathcal{V}$ and $\mathcal{E}$ are respectively the sets of nodes and edges of $\mathcal{G}$, and can only communicate with their neighbours in $\mathcal{G}$, as shown in Figure 1.3.2. Besides, each node maintains a local copy of the parameter $x_t(i)$, whereas there was a global x_t computed by the server and broadcast to all nodes with centralized algorithms. In the centralized setting, and although other actions are possible, the communication step actually only consisted in a

global averaging of the gradients. In the same way, in this decentralized setting, communications are generally abstracted as a multiplication by a so-called "gossip matrix" $W \in \mathbb{R}^{n\times n}$, which performs a partial averaging. Indeed, full averaging is not possible with only one round of pairwise communications. Such algorithms are called *gossip algorithms* [Boyd et al., 2006, Nedic and Ozdaglar, 2009, Shi et al., 2015a, Nedic et al., 2017], and the gossip matrix W satisfies the following conditions:

(1) W is an $n \times n$ symmetric positive semi-definite matrix,
(2) The kernel of W is such that: $\mathrm{Ker}(W) = \mathrm{Span}(\mathbb{1})$, where $\mathbb{1} = (1, \dots, 1)^\top$,
(3) W is such that $W_{ij} = 0$ if i and j are not neighbours, *i.e.*, $(i, j) \notin \mathcal{E}$.

Because there is no central point of failure, decentralized algorithms are generally more robust than their centralized counterparts [Lian et al., 2017a]. In this paradigm, nodes can perform two distinct actions to update their parameter:

- *Local computations*: query their local oracles (for instance to compute $\nabla f_i(x_t(i))$),
- *Local communications*: communicate with their neighbours using matrix W.

One could hope to adapt gradient descent in the same way as it was adapted for centralized methods, and perform the following iterations:

$$x_{t+1} = (I - W)x_t - \eta_t G(x_t), \tag{1.3.6}$$

where $x_t \in \mathbb{R}^{n\times d}$ and G_t is such that $(G(x_t))_i = \nabla f_i(x_t(i))$. This is for instance the form of the distributed subgradient methods D-PSGD and its stochastic and asynchronous variants [Nedic and Ozdaglar, 2009, Ram et al., 2009, 2010], as well as other extensions [Lian et al., 2017b, Tang et al., 2018b]. Yet, these iterations do not converge in general to $x_\star \otimes \mathbb{1}$, where $x_\star$ the solution of Problem (1.3.1) and $\otimes$ is the Kronecker product. Indeed, $\nabla f_i(x_\star) \neq 0$ in general so $G(x_\star) \neq 0$ and $x_\star$ is not a fixed point of (1.3.6). In order to recover a provable convergence rate, to the global optimum, diminishing step-sizes must be used, which slow convergence down drastically even in the favorable smooth and strongly convex case. Thus, more care is required when building decentralized algorithms than in the centralized setting. We detail the main approaches and algorithms in Section 1.4, and refer the interested reader to these surveys [Sayed, 2014, Nedic, 2020, Gorbunov et al., 2020] for more general introductions to these algorithms. Besides the pure optimization setting, decentralized algorithms, and in particular gossip protocols also guarantee some level of anonymity, as we will see in Chapter 6.

1.3.3. Local methods and federated learning. The term *Federated Learning* [Kairouz et al., 2019, McMahan et al., 2021] has gained a lot of attention in the past few years, and refers to a subfield of distributed optimization that focuses on *centralized* architectures, with *decentralized* data. Although there are many flavours of federated learning, it is not very different from the setting presented in Section 1.3.1, since the data generally stays decentralized: otherwise algorithms would consist in sending the data to the server, performing computations, and then sending it back to the workers, which is trivial in term of parallelism. Thus, in this paragraph, we will discuss *local methods* [Stich, 2018, Lin et al., 2019, Karimireddy et al., 2020, Gorbunov et al., 2021], which are often what Federated Learning algorithms are about. In particular, we focus on local SGD, (also called FedAvg [Konečnỳ et al., 2016]), which consists in the following steps:

(1) The server broadcasts x_t to all workers

(2) Worker i sets $x_{t,0}(i) = x_t$, and for $k = 1$ to K, performs $x_{t,k}(i) = x_{t,k-1}(i) - \eta_{t,k} g_{t,k}(i)$, with $\mathbb{E}[g_{t,k}(i)] = \nabla f_i(x_{t,k-1}(i))$.
(3) All workers send $x_{t,K}$ to the server, which computes $x_{t+1} = \frac{1}{n}\sum_{i=1}^n x_{t,K}(i)$.

Informally, each node performs a standard SGD algorithm locally, and periodic averaging are performed every K iterations to control the drift between the local parameters of the different nodes. Thus, this local SGD method (and local methods in general) interpolates between the centralized and decentralized settings. Indeed,

- Each node has its own parameter, and updates it by performing local computations, just like the decentralized setting.
- Nodes have access to a global averaging operation, which allows them to periodically restart from a shared parameter, just like in the centralized setting.

Of course, there are many variants of local methods, and they do not all fall into this category, but this midpoint interpretation between centralized and decentralized methods still holds most of the time. In particular, results can be transfered from one setting to the other: one can use a decentralized algorithm with a full averaging gossip matrix ($W = I - \frac{1}{n}\mathbb{1}\mathbb{1}^\top$) and obtain a local method, and one can replace the exact averaging step by an approximate one (using $x_{t+1} = (I - W)x_{t,K}$ instead of $x_{t+1} = \mathbb{1}\mathbb{1}^\top x_{t,K}/n$) and obtain a decentralized algorithm. The other difference is about the algorithms that are considered: local methods often consider SGD-like algorithms, in which all nodes draw stochastic gradients from the same function. On the other hand, the study of decentralized algorithms often has a strong focus on the dependence on the spectral gap of the gossip matrix, a quantity of interest that does not appear in the study of local methods. In particular, the algorithms presented in Chapters 3 and 4 can be seen as local methods since they alternate between a series of local computation steps and (partial) averaging operations through gossip communications. In summary, *Federated Learning* designates a wide variety of settings which are not (very) different from standard centralized optimization, and the local methods used in this setting can be seen as a midpoint between centralized and decentralized optimization, since there is a global aggregator, the server, but each node has its own local parameter.

1.3.4. Asynchronous methods. So far and in particular when presenting centralized algorithms in Section 1.3.1, we have assumed the communication time τ to be constant. Yet, a more precise model would consider that each node i has a specific delay τ_i associated with it, and in this case the time taken by centralized gradient descent would be:

$$T^{\mathrm{AGD}}_{\mathrm{centralized}}(\varepsilon) = O((1 + \max_i \tau_i)\sqrt{\kappa}\log(\varepsilon^{-1})). \tag{1.3.7}$$

In real systems, communication times may be rather heterogeneous, and yet in a synchronous implementation of gradient descent, all nodes need to wait for $\tau_{\max}$ before they start computing again. In this case, the slowest node sets the pace of the whole system, and so it can actually be beneficial to remove nodes (and thus computing power), so that the other nodes don't wait so long. This is called the *straggler* problem. Note that if the delays τ_i are stochastic, then $\mathbb{E}[\max_i \tau_i]$ can still be quite large even though all τ_i have small and comparable means. Thus, asynchronous methods rely on aggregating small updates as soon as they are available instead. For instance, asynchronous SGD would perform:

$$x_{t+1} = x_t - \eta_t \nabla f_i(x_{t-\tau_i(t)}), \tag{1.3.8}$$

where the delays $\tau_i(t)$ account for the fact that the parameter at the server has been updated between the time the server sends the parameter to the node, and the time the node finishes its local computation and sends the gradient back. With these asynchronous methods, the updates are less efficient since they rely on old gradients, but much faster since no waiting is required. There is a very large literature on asynchronous methods, ranging from HogWild (Asynchronous SGD) [Recht et al., 2011, Mania et al., 2017], to asynchronous coordinate descent [Liu et al., 2015, Cheung et al., 2021, 2020] and asynchronous versions of variance-reduced algorithms [Reddi et al., 2015, Leblond et al., 2017]. Yet, the focus of this thesis is not on asynchronous methods, and we refer the interested reader to Assran et al. [2020]. Note that we mention "asynchronous gossip" at some point in Chapters 2 and 3, yet, this refers to randomized gossip algorithms (local pairwise interactions) as opposed to synchronous gossip algorithms, and not to asynchronous algorithms as presented in this section.

1.4. Standard Decentralized Approaches with Linear Convergence

We have briefly presented the decentralized setting in Section 1.3.2, and highlighted that decentralized algorithms cannot be derived in a straightforward way from their single-machine counterparts the way it can often be done with centralized algorithms. In this section we introduce the basic approaches to obtain decentralized algorithms in the standard smooth and strongly convex setting. We start by discussing the averaging problem [Shah, 2009], and then how they can be extended to minimize more general objectives. Then, we introduce the dual approach, which is at the heart of this thesis, and finally discuss other decentralized related works.

1.4.1. The averaging problem. In the averaging problems, the n nodes of the network possess a local value $c_i \in \mathbb{R}^d$, and the goal is to compute $\bar{c} = \frac{1}{n}\sum_{i=1}^n c_i$. This is equivalent to Problem (1.3.1) with quadratic local functions $f_i(x) = \frac{1}{2}\|x - c_i\|^2$. Decentralized averaging 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]. The simplest gossip algorithm to solve this problem is to simply iterate, with $x_t \in \mathbb{R}^{n\times d}$ the concatenation of all local parameters:

$$x_{t+1} = \left(I - \frac{1}{\lambda_{\max}(W)} W\right) x_t. \tag{1.4.1}$$

If we denote $\gamma \in [0, 1]$ the spectral gap of the matrix W, which is defined as the ratio between the smallest non-zero eigenvalue and the largest eigenvalue of W, *i.e.*, $\gamma = \lambda_{\min}^+(W)/\lambda_{\max}(W)$, then a very simple proof leads to:

$$\|x_{t+1} - \bar{c}\mathbb{1}\|^2 \leq (1-\gamma)^t \|x_0 - \bar{c}\mathbb{1}\|^2. \tag{1.4.2}$$

This highlights the role of γ as a central quantity in gossip algorithms, which will be the case for all subsequent decentralized algorithms. This quantity depends on the size of the network, and it is of order $O(1)$ for the complete graph $O(n^{-1})$ for the 2D grid, and $O(n^{-2})$ for the ring and line graphs [Mohar, 1997]. More generally, the eigengap of a graph is linked with its diameter Δ by $\gamma^{-1/2} \geq \Delta/(2\sqrt{2}\log_2(n))$ for regular networks [Alon and Milman, 1985].

It is possible to reduce the dependence on γ by multiplying by $P(W)$, with P a given polynomial, instead of W. If P is of degree k, this consists in performing k communications steps at once. In particular, a dependence on $\gamma^{1/2}$ can be obtained when P is chosen as the Chebyshev polynomial of degree $\lceil \gamma^{-1/2} \rceil$ [Oreshkin et al., 2010], since in this case $P(W)$ has eigengap $O(1)$. This is called *Chebyshev* acceleration, and it focuses only on improving the dependency on the spectral gap (worst-case dependence). Faster methods (in the transitory regime) that focus on the whole spectrum can be tailored to specific graphs by using other families of orthogonal polynomials [Berthier et al., 2020].

Yet, the multiplication by W performed by the iterations (1.4.1) require global synchronization: all nodes communicate with their neighbours at the same time, and then once this is finished they repeat the operation. Instead, *Randomized gossip algorithms* [Boyd et al., 2006] perform a series of pairwise averaging as follows:

$$x_{t+1}(i) = x_{t+1}(j) = \frac{x_t(i) + x_t(j)}{2}, \text{ with probability } p_{ij}. \tag{1.4.3}$$

This simple algorithm achieves the same $O(\gamma^{-1})$ asymptotic rate as (1.4.1), but relying on random activations of edges only, instead of global synchronous communication. Yet, acceleration based on orthogonal polynomials cannot be used in this case, and methods based on shift registers [Cao et al., 2006] or heavy-ball acceleration [Loizou and Richtárik, 2018] only show improved convergence rates for expected iterate, which is equal to the one that would be obtained by a synchronous method, and do not control the variance. Before the results in this thesis, only ad-hoc accelerated methods have been developed for specific graphs such as geographic gossip [Dimakis et al., 2010]. In Chapter 2, we show how the dual approach specified to the averaging problem leads to an actual accelerated randomized gossip algorithm with improved guarantee. Subsequent work obtains comparable results using the randomized Kaczmarz algorithm [Loizou et al., 2019], but their results only apply to the averaging problem, and do not extend to general convex optimization.

1.4.2. Linearly converging algorithms. In order to get rid of the diminishing step-sizes from distributed subgradient methods and accelerate the convergence, two main direct approaches exist: ad-hoc corrections of the iterations fixed-point to guarantee convergence to the global optimum, and gradient tracking. We present these 2 approaches in this Section, and leave the dual approach, which is at the heart of this thesis, to Section 1.4.3.

EXTRA [Shi et al., 2015a]-type methods. The main idea leading to the EXTRA [Shi et al., 2015a,b] is that the naive implementation from (1.3.6) does not converge to the global optimum in general because the gradients do not mix, and $G(x_\star) \neq 0$ in general. To cancel this remaining error, EXTRA writes the steps from (1.3.6) for two consecutive timesteps with two different gossip matrices W and $\tilde{W}$:

$$\begin{aligned} x_{t+2} &= (I - \tilde{W})x_{t+1} - \eta G(x_{t+1}), \\ x_{t+1} &= (I - W)x_t - \eta G(x_t), \end{aligned}$$

and then perform the second-order recursion given by the difference of the two. One can easily verify that the limit point of these iterations is consensual ($Wx_\star = 0$) and minimizes the global problem ($\mathbb{1}^\top G(x_\star) = 0$). Additionally, it is possible to prove that EXTRA converges linearly to this global optimum. Extensions of EXTRA include, *e.g.*, Li et al. [2019], which

includes support for a separable non-smooth term. Tight convergence rates were then given in Li and Lin [2020], together with a generic Catalyst [Lin et al., 2015a] acceleration.

Gradient Tracking. Instead of directly correcting the fixed point of the iterations like EXTRA does, *gradient tracking* approaches are based on the fact that one would like to (conceptually) apply the following iterations:

$$x_{t+1} = (I - W)x_t - \frac{\eta_t}{n}\mathbb{1}\mathbb{1}^\top G(x_t). \tag{1.4.4}$$

Yet, the global average gradient $\mathbb{1}^\top G(x_t)/n$ cannot be computed in a decentralized way, so it is approximated, or *tracked* by an auxiliary variable y_t instead:

$$y_{t+1} = (I - W)y_t + G(x_{t+1}) - G(x_t). \tag{1.4.5}$$

This idea was introduced independently by Xu et al. [2015] with the AUG-DGM algorithm, and by Di Lorenzo and Scutari [2015] with the NEXT algorithm for the non-convex setting. It is also at the heart of the DIGing algorithm [Nedic et al., 2017], which shows linear convergence in the smooth and strongly convex setting, and supports time-varying and directed digraphs. Gradient tracking is still a fruitful algorithm design principle, and is at the heart of Sun et al. [2019b], Pu and Nedić [2020], Xin et al. [2020b]. It turns out that the two previous approaches are very similar, and can actually be analyzed within the same framework. This has been done by Jakovetić [2018] and by Xu et al. [2020c,b] in the composite setting, along with tight convergence rates. More recently, Li and Lin [2020], Li et al. [2020c] also contributed to the theory of these algorithms, and used it to develop accelerated and variance-reduced versions, as we will see in the next sections.

1.4.3. The dual approach. In this section, we present the dual approach to building decentralized optimization algorithms. We present it in more details than the previous ones since Chapters 2, 3 and 4 build on this approach. More specifically, in the standard dual approach [Boyd et al., 2011, Jakovetić et al., 2014], we rewrite Problem (1.3.1) as:

$$\begin{aligned} &\min_{x\in\mathbb{R}^{n\times d}} \sum_{i=1}^n f_i(x_i). \\ &\text{such that } x_i = x_j \ \forall (i,j) \in \mathcal{E}. \end{aligned} \tag{1.4.6}$$

Using Lagrangian duality in the same way as in Section 1.2.2, we obtain that this problem is equivalent to the following dual problem:

$$\min_{y\in\mathbb{R}^{E\times d}} \sum_{i=1}^n f_i^*(e_i^\top A y), \tag{1.4.7}$$

where E is the number of edges in the graph $\mathcal{G}$ and $A \in \mathbb{R}^{n\times E}$ is such that $e_{ij}^\top A^\top x = \mu_{ij}(x_i - x_j)$. Lagrangian duality allows us to transform the constrained problem (1.4.6) into an unconstrained problem. Yet, there is no decentralized interpretation at the moment. The gradient descent iterations for the dual problem write:

$$y_{t+1} = y_t - A^\top \nabla F^*(z_t), \tag{1.4.8}$$

where $F^* : x \mapsto \sum_{i=1}^n f_i^*(e_i^\top x)$ for $x \in \mathbb{R}^{n\times d}$. In particular, with $z_t = Ay_t \in \mathbb{R}^{n\times d}$, these iterations rewrite:

$$z_{t+1} = z_t - AA^\top \nabla F^*(z_t). \tag{1.4.9}$$

From here, two main observations can be made:

- The dual gradient $\nabla F^*(z_t)$ is such that $(\nabla F^*(z_t))_i = \nabla f_i^*((z_t)_i)$ can be computed locally at node i.
- The matrix $AA^\top \in \mathbb{R}^{n\times n}$ is a gossip matrix on the graph $\mathcal{G}$.

This justifies taking the notations $W = AA^\top$ and $\Theta_t = \nabla F^*(z_t)$, and gradient descent on the dual problem can thus be written as:

$$\begin{aligned} \Theta_t &= \nabla F^*(z_t) \\ z_{t+1} &= z_t - W\Theta_t. \end{aligned} \tag{1.4.10}$$

The primal-dual optimality conditions imply that if $x_\star$ and $y_\star$ are respectively the primal and dual solutions, then for all $i \in \{1, \dots, n\}$,

$$\nabla f_i(x_\star) = e_i^\top A y_\star, \text{ so } x_\star = \nabla f_i^*(e_i^\top A y_\star). \tag{1.4.11}$$

In particular, $(\Theta_t)_i$ converges to the global minimizer $x_\star$ for all i. Thus, the algorithm defined by (1.4.10) is a valid decentralized algorithm that converges to the global minimizer. Its rate of convergence is given by the condition number of the dual function, which can directly be upper bounded by κ_l/γ (the condition number of the individual f_i^*, which is the same as the f_i, times the condition number of $AA^\top$, which is equal to γ^{-1}).

Thus, the dual approach allows to design and analyze decentralized algorithms in a principled way. We have shown the algorithm that we obtain by using gradient descent on the dual problem, but other algorithms can be used to obtain meaningful results in a wide variety of settings. In particular Scaman et al. [2017b] apply accelerated gradient descent (and thus obtained an accelerated algorithm), whereas Maros and Jaldén [2018] apply standard dual gradient ascent, but modify the iterates to handle time-varying directed graphs. ADMM-type [Boyd et al., 2011, Jakovetic et al., 2011, Shi et al., 2014] or inexact augmented Lagrangian methods [Jakovetić et al., 2014] can also be used on the primal-dual formulation of the problem to obtain linearly converging decentralized algorithms.

Another interesting feature is that using Lagrangian duality, primal constraints become dual variables. Since there is one constraint per edge in the primal problem, then there is one variable per edge in the dual problem. In particular, pairwise algorithms can be recovered directly by applying coordinate descent on the dual problem instead of batch gradient descent. This observation is at the heart of Chapter 2.

Besides, one can notice that the structure of the problem and the derivations are very similar to the dual approach for finite-sum problems, in particular when using coordinate descent. A natural question in this case is thus whether it is possible to obtain a decentralized variance-reduced algorithm by combining both approaches into the same problem. Chapter 3 answers this question positively, and Chapter 4 focuses on obtaining a dual-free version of this algorithm.

1.4.4. Related settings. We have mainly discussed methods that focus on the smooth and strongly convex setting, which is the one that we mainly study in this thesis. Yet, decentralized optimization considers a wide variety of problems with a wide variety of approaches. We mention in this Section the main ones that were not previously discussed.

Composite setting. A widespread problem is that of composite optimization, which is when the objective is of the form $f+g$, where f is a smooth and strongly convex function, and g is a non-smooth function but for which the proximal operator is known. This corresponds for instance to ℓ_1-regularized problems, which are very common to obtain sparse models, or

in computer vision. Decentralized algorithms that consider such setting include (but are not restricted to) Shi et al. [2015b], Wang et al. [2018a], Li et al. [2019], Xu et al. [2020b,c].

Non-convex setting. A significant effort has also been dedicated to the non-convex setting, and in particular for gradient-tracking algorithms [Di Lorenzo and Scutari, 2015, Tian et al., 2018, Cannelli et al., 2020]. There is also another line of work, motivated mainly by deep learning application, that studies variants of the D-PSGD algorithm [Lian et al., 2017b, Assran et al., 2019, Tang et al., 2018b]. An analysis for the standard decentralized SGD algorithm is also provided in [Koloskova et al., 2020].

Penalty method. The penalty method consists in replacing the hard consensus constraint in the primal problem (1.4.6) by a soft penalization instead. Thus, these methods do not converge to the exact global minimizer, but they are much simpler to design, as one can then use standard convex optimization algorithms on the penalized problem [Dvinskikh and Gasnikov, 2021, Li et al., 2020b, Rogozin and Gasnikov, 2020].

Compressed Communications. In modern machine learning applications, the dimension of the models that we wish to optimize can be very big, and sending gradients can thus be very expensive. To tackle this problem, a line of work focuses on using compressed communications without degrading the performance (too much) [Tang et al., 2018a, Koloskova et al., 2019a, 2020, Kovalev et al., 2021]

Time-varying graphs. Most of the methods discussed earlier focus on fixed undirected networks. Yet, many real-life networks actually vary with time, and it is desirable to design algorithms that are robust to this. Yet, the "time-varying" setting is large, and encompasses many different assumptions. In particular, Rogozin et al. [2019] develop optimal algorithms for graphs that vary slowly, while Sun et al. [2016], Koloskova et al. [2020] consider graphs that may not be connected individually, but for which the composition of τ successive graphs is (in expectation). In this work, and in particular in Chapter 2, we employ randomized communications over a fixed graph, which can be seen as a special (restricted) case of time-varying graphs.

Directed graphs. Undirected communications require some level of synchronization between agents, and can be quite challenging to implement in practice. Instead, communicating over directed graphs allows nodes to *push* information to their neighbours in a much simpler way. Yet, analyses are much more challenging, and in particular the dual approach is more challenging to leverage in this case. We refer the interested reader to Kempe et al. [2003b], Nedić and Olshevsky [2016], Sun et al. [2016], Assran et al. [2019] for more details on these methods.

1.4.5. Optimal Decentralized Methods. So far, we have presented the main approaches that can be used to obtain decentralized algorithms, but without discussing the main refinements that we presented in Section 1.2, *i.e.*, acceleration and variance reduction. We present accelerated decentralized optimization results in this section. We have seen in Section 1.4.1 that acceleration for the synchronous gossip problem could be handled using Chebyshev acceleration. Yet, the problem is much harder for general convex objectives, as we now show. Similarly to the single-machine setting, we first introduce a lower bound and then discuss the optimality of the methods.

THEOREM 9. *[Scaman et al., 2017b, Corollary 2] For any $\gamma > 0$, there exists a gossip matrix W of eigengap γ and σ-strongly convex and L-smooth functions such that, with $\kappa_\ell =$*

β/α, for any black-box procedure using W, the time to reach a precision $\varepsilon > 0$ is lower bounded by:

$$\Omega\left(\sqrt{\kappa_l}\left(1+\frac{\tau}{\sqrt{\gamma}}\right)\ln(\varepsilon^{-1})\right). \tag{1.4.12}$$

In this case, black-box procedures are similar to the ones considered in 6, with the difference that $(x_{t+1})_i$ can only be computed using first-order oracles of the *local* function f_i, the history $(x_k)_i$ for $k \leq t$ and $(x_t)_j$ if j is a neighbour of i. We recall that local computations take time 1, whereas communications take time τ. Although the initial convergence results on EXTRA are rather weak, Li and Lin [2020] prove that the iteration complexity to reach precision ε with EXTRA is of order

$$O\left((1+\tau)\left(\kappa_l+\gamma^{-1}\right)\log(\varepsilon^{-1})\right). \tag{1.4.13}$$

Surprisingly, these rates completely separate the influence of the graph (γ) and of the local functions (κ_l), and they are optimal in the special cases in which $\kappa_l = O(\gamma^{-1})$. Yet, they are not optimal in general and, similarly to convex optimization, optimal methods are obtained using acceleration. In particular, after introducing the lower bound, Scaman et al. [2017b] introduce SSDA, which consists in applying accelerated gradient descent to the dual problem, and which converges as:

$$O\left((1+\tau)\sqrt{\frac{\kappa_l}{\gamma}}\ln(\varepsilon^{-1})\right). \tag{1.4.14}$$

This obtains the right dependence in the condition number, but does not exactly match the lower bound from Equation (1.4.12). In particular, the spectral gap of the graph still plays a role in the convergence rate, even when considering arbitrarily fast communications ($\tau \to 0$). Yet, there is a simple fix in this case, which is to use Chebyshev acceleration just like in the gossip averaging setting. This leads to the MSDA algorithm, for which the convergence guarantee exactly matches the lower bound from (1.4.12). We refer to [Uribe et al., 2020] for more details on the dual approach for optimal decentralized algorithms in a wide variety of settings.

Yet, this is not the end of the story, as MSDA requires *dual* oracles, which are generally much more expensive to compute than primal ones. Direct approaches relying on the combination of gradient tracking and Nesterov acceleration were developped [Qu and Li, 2019], but do not obtain the expected $\sqrt{\kappa}$ dependence. Other approaches based on the penalty method [Li et al., 2020b, Rogozin and Gasnikov, 2020] were developed, but they do not converge to the true minimizer unless a decreasing penalty is used. Another line of work consists in mimicking approximate centralized gradient descent by employing accelerated methods from Section 1.4.1 to mix the gradients [Ye et al., 2020]. Yet, similarly to the penalty method with decreasing step-size, these methods pay extra logarithmic factors in their convergence rates. Arjevani et al. [2020] propose an inexact augmented Lagrangian framework that generalizes MSDA, but still require computing the proximal operator of local functions. A catalyst (generic) acceleration of the EXTRA was developed in Li and Lin [2020], but again due to the generic nature of acceleration then extra logarithmic factors and hyperparameter tuning need to be paid. The first primal accelerated decentralized method was obtained by Kovalev et al. [2020], using a primal-dual approach together with Nesterov acceleration. Equivalent

acceleration in the non strongly convex setting ($\sigma = 0$) were obtained independently using a primal-dual approach as well [Xu et al., 2020a].

In order to finish the extension of the convex optimization results presented in Section 1.2 to the decentralized setting, we would need to present (accelerated) variance-reduced methods. Yet, this is the focus of Chapters 3 and 4, and the main other references that address the same problem were published either concurrently of after the articles that correspond to these chapters. Thus, we leave these discussions for the corresponding chapters.

CHAPTER 2

Accelerated Decentralized Optimization with Pairwise Updates

In this Chapter, 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., 2017b], 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.

This Chapter is based on the paper *Accelerated Decentralized Optimization with Local Updates for Smooth and Strongly Convex Objectives* [Hendrikx, Bach, and Massoulié, 2019a], published at AISTATS 2019.

2.1. 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., 2012b, Shi et al., 2015a, Mokhtari and Ribeiro, 2016, Scaman et al., 2017b, 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., 2018b].

A popular way to make first order optimization faster is to use Nesterov acceleration [Nesterov, 2013c]. Accelerated gradient descent in a dual formulation yields optimal synchronous algorithms in the decentralized setting [Scaman et al., 2017b, Ghadimi et al., 2013]. Variants of accelerated gradient descent include the acceleration of the coordinate descent algorithm [Nesterov, 2012, Allen-Zhu et al., 2016, Nesterov and Stich, 2017], that we use in this paper to solve the problem in Scaman et al. [2017b]. 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., 2017b] even in idealized homogeneous settings in which nodes never wait when performing synchronous algorithms. In terms of efficiency, we match the oracle complexity of SSDA with lower communication cost. This extends a result that is well-known in the case of averaging, *i.e.*, that randomized gossip algorithms match the rate of synchronous ones [Boyd et al., 2006]. We also exhibit a clear experimental speedup when the distributions of nodes computing power and local smoothnesses have a high variance.

Choosing quadratic f_i functions leads to solving the distributed average consensus problem, in which each node has a variable c_i and for which the goal is to find the mean of all variables $\bar{c} = \frac{1}{n}\sum_{i=1}^{n} c_i$. It is a historical problem [DeGroot, 1974, Chatterjee and Seneta, 1977] that still attracts attention [Cao et al., 2006, Boyd et al., 2006, Loizou and Richtárik, 2018] with many applications for averaging measurements in sensor networks [Xiao et al., 2005] or load balancing [Diekmann et al., 1999]. Fast synchronous algorithms to solve this problem exist [Oreshkin et al., 2010] but no asynchronous algorithms match their rates. We

show that ESDACD is faster at solving distributed average consensus than standard asynchronous approaches [Boyd et al., 2006, Cao et al., 2006] as well as more recent ones [Loizou and Richtárik, 2018] that do not show improved convergence rates for the second moment of the error. The complexity of gossip algorithms generally depends on the smallest non-zero eigenvalue of the gossip matrix W, a symmetric semi-definite positive matrix of size $n \times n$ ruling how nodes aggregate the values of their neighbors such that $\text{Ker}(W) = \text{Vec}(\mathbb{1})$ where $\mathbb{1}$ is the constant vector. We improve the rate from $\lambda_{\min}^+(W)$ to $O\left(\frac{1}{\sqrt{n}}\sqrt{\lambda_{\min}^+(W)}\right)$ where $\lambda_{\min}^+(W) \leq \frac{1}{n-1}$ 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].

2.2. 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 $x, y \in \mathbb{R}^d$:

$$f_i(x) - f_i(y) \leq \nabla f_i(y)^T(x-y) + \frac{L_i}{2}\|x-y\|^2 \tag{2.2.1}$$

$$f_i(x) - f_i(y) \geq \nabla f_i(y)^T(x-y) + \frac{\sigma_i}{2}\|x-y\|^2. \tag{2.2.2}$$

Note that the Fenchel conjugate f_i^* of f_i (defined in Equation (2.3.5)) 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:

$$\min_{x \in \mathbb{R}^{n \times d}:\ x_i = x_j\ \forall i,j \in \{1,\dots,n\}} F(x). \tag{2.2.3}$$

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.

2.3. 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.

2.3.1. Problem derivation. In order to obtain the algorithm, we consider a matrix $A \in \mathbb{R}^{n \times E}$ such that $\mathrm{Ker}(A^T) = \mathrm{Vec}(\mathbb{1})$ 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 (2.2.3) can be expressed as $A^T x = 0$ because if $x \in \mathrm{Ker}(A^T)$ then all its coordinates are equal and the problem writes:

$$\min_{x \in \mathbb{R}^{n \times d}:\ A^T x = 0} F(x). \tag{2.3.1}$$

This problem is equivalent to the following one:

$$\min_{x \in \mathbb{R}^{n \times d}} \max_{\lambda \in \mathbb{R}^{E \times d}} F(x) - \langle \lambda, A^T x \rangle, \tag{2.3.2}$$

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

$$\max_{\lambda \in \mathbb{R}^{E \times d}} \min_{x \in \mathbb{R}^{n \times d}} F(x) - \langle A\lambda, x \rangle \tag{2.3.3}$$

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

$$\max_{\lambda \in \mathbb{R}^{E \times d}} -F^*(A\lambda), \tag{2.3.4}$$

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

$$F^*(y) = \max_{x \in \mathbb{R}^{n \times d}} \langle x, y \rangle - F(x). \tag{2.3.5}$$

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 $\mu_{ij} \in \mathbb{R}$:

$$Ae_{ij} = \mu_{ij}(e_i - e_j). \tag{2.3.6}$$

This choice of A satisfies $e_{ij}^T A^T \mathbb{1} = 0$ for all $(i, j) \in E$ and $\mathrm{Ker}(A^T) \subset \mathrm{Vec}(\mathbb{1})$ 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.

2.3.2. 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

$$\theta^2 = \min_{ij} \frac{p_{ij}^2}{\mu_{ij}^2 e_{ij}^T A^+ A e_{ij}} \frac{\sigma_A}{\sigma_i^{-1} + \sigma_j^{-1}} \geq \frac{\sigma_A}{S^2}. \tag{2.3.7}$$

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

$$g_{ij}(y_t) = e_{ij} e_{ij}^T A^T \nabla F^*(Ay_t) \in \mathbb{R}^{E \times d}. \tag{2.3.8}$$

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}
  $y_{t+1} = (1 - \delta)y_t + \delta v_t - \eta_{ij} g_{ij}(y_t)$
  $v_{t+1} = (1 - \theta)v_t + \theta y_t - \frac{\theta}{\sigma_A p_{ij}} g_{ij}(y_t)$

THEOREM 10. *Let y_t and v_t be the sequences generated by Algorithm 1. Then:*

$$2\left(\mathbb{E}[F_A^*(x_t)] - F_A^*(x^*)\right) + \sigma_A \mathbb{E}[r_t^2] \leq C(1 - \theta)^t, \tag{2.3.9}$$

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 10 shows that Algorithm 1 converges with rate θ. Lemma 6, in Appendix 2.C shows that

$$\sigma_A \geq \frac{\lambda_{\min}^+(A^T A)}{L_{\max}}, \tag{2.3.10}$$

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 $\lambda_{\min}^+(A^T A)$ by the same constant. The $p_{ij}^2/(\sigma_i^{-1} + \sigma_j^{-1})$ term indicates that non-smooth edges should be sampled more often, and the square root dependency is consistent with known results for accelerated coordinate descent methods [Allen-Zhu et al., 2016, Nesterov and Stich, 2017]. If both sampling probabilities and smoothnesses are fixed, the 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 $\|v_{t+1} - x^*\|_{A^+A}^2$ and follows the same scheme as by Nesterov and Stich [2017]. However, F_A^* is not strongly convex because matrix $A^T A$ is generally not full rank. Yet, F_A^* is strongly convex for the pseudo-norm 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 2.C. □

2.3.3. 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 $s_{ij} = \begin{pmatrix} \frac{\theta\mu_{ij}^2}{p_{ij}\sigma_A} & \mu_{ij}^2\eta_{ij} \end{pmatrix}^T$.

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**
    $\begin{pmatrix} v_t(r)^T \\ y_t(r)^T \end{pmatrix}_r = B^{t-t_r} \begin{pmatrix} v_{t_r}(r)^T \\ y_{t_r}(r)^T \end{pmatrix}$
    $z_r = \nabla f_r^*(y_t(r))$
    $send\_gradient(x_r, j)$ // non blocking
    $z_{dist} = receive\_gradient(j)$ // blocking
    $g_t(r) = s_e\,(z_r - z_{dist})$
    $\begin{pmatrix} v_{t+1}(r)^T \\ y_{t+1}(r)^T \end{pmatrix}_r = B \begin{pmatrix} v_t(r)^T \\ y_t(r)^T \end{pmatrix} - g_t(r)^T$
    $t_r = t + 1$
  t = t + 1
return z_r

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

2.3.4. 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.

2.4. Performances

2.4.1. 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 $\mathcal{G}$ is omitted when clear from the context.

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 2.3.4 may drastically slow the algorithm down, making it inefficient to use cheaper iterations. Theorem 11 shows that this does not happen for regular graphs with homogeneous probabilities. We note $\tau_{\max}$ the maximum delay of all edges.

THEOREM 11. *If we denote $T_{\max}(k)$ the time taken by ESDACD to perform k iterations when edges are sampled according to the distribution p:*

$$\bar{\tau} = \mathbb{E}\left[\frac{1}{k} T_{\max}(k)\right] \leq c\bar{p}\tau_{\max} \tag{2.4.1}$$

with a constant $c < 14$.

The proof of Theorem 11 is in Appendix 2.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. We now introduce an assumption on the degree of the graphs.

ASSUMPTION 2. *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 2 is satisfied for many standard graphs and probability distribution over edges. In particular, it is satisfied by the uniform distribution for regular degree graphs. It can be used to show the following corollary:

COROLLARY 1. *If $\mathcal{G}$ satisfies Assumption 2 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:*

$$\mathbb{E}\left[T_{\max}(k)\right] \leq c\frac{\tau_{\max}}{n} k + o(k). \tag{2.4.2}$$

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 2, *e.g.*, star networks. These topologies are more suited to centralized algorithms because some nodes take part in almost all updates.

2.4.3. 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:

$$\mathcal{E}_2(t) \leq (1 - \theta_{\text{gossip}})^{2t}\mathcal{E}_2(0), \tag{2.4.3}$$

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]. We now introduce the following assumption, is useful to analyze accelerated randomized gossip.

Assumption 3. *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 (2.3.6) with $\mu_{ij} = 1$ and uniquely defines $\mathcal{G}(n)$.*

This 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 3 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 3.

Corollary 2. *If $\mathcal{G}$ satisfies Assumption 3 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:*

$$\theta_{ESDACD} \geq \frac{c}{\sqrt{n}}\sqrt{\theta_{\text{gossip}}}. \tag{2.4.4}$$

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.

2.4.4. Comparison to SSDA. The results described in Theorem 10 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 2 and 3. There exists $c > 0$ such that:*

$$\frac{\theta_{ESDACD}}{\bar{\tau}_{ESDACD}} \geq c \frac{1}{\tau_{\max}} \sqrt{\frac{\gamma}{\kappa}} = c \, \frac{\theta_{SSDA}}{\bar{\tau}_{SSDA}}, \tag{2.4.5}$$

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 (2.3.6).

The proof is in Appendix 2.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.

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.

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.

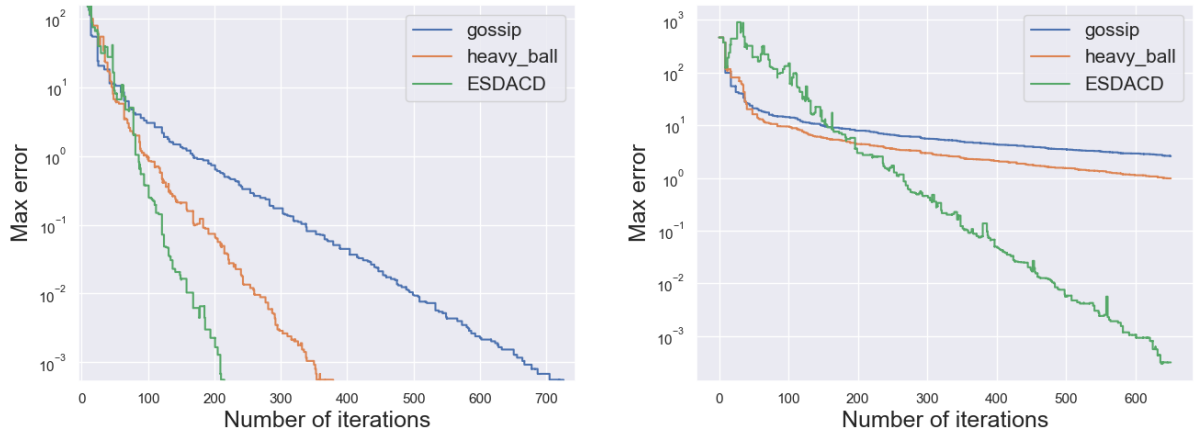
2.5. Experiments

2.5.1. 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(a) and 2(b) 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 2(b) 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 2.C as in Nesterov and Stich [2017] would lead to a more complex algorithm with better initial performances.

2.5.2. 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. [2017b]:

- 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



(a) 10×10 grid. (b) Ring graph of size 100.

FIGURE 2. ESDACD, pairwise gossip and heavy ball gossip on various graphs

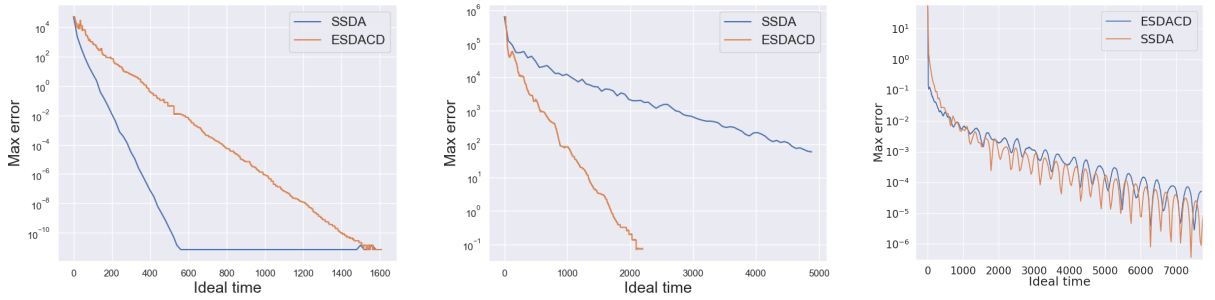
the first class and 1 for the second class. Classes are balanced. At each node, the loss function is $f_i(\theta_i) = \sum_{j=1}^{N} \ln\left(1 + \exp^{-y_{i,j} X_{i,j}^T \theta}\right) + c_i \|\theta\|^2$ with $c_i = 1$.

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 $\max_i F(\theta_i) - \min_x F(x)$. 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 3(a) 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 3(b) shows that ESDACD is computationally more efficient than SSDA on the regression problem because it has a far lower final error although it uses 2 times less gradients. This can be explained by larger step sizes along regular edges and suggests that ESDACD adapts more easily to changes in local regularity, even with uniform sampling probabilities. ESDACD is also much faster since in average, each node performs 2 iterations in half the time needed for one SSDA iteration. For the classification problem, strong convexity is more homogeneous because it only comes



(a) Homogeneous regression problem (b) Heterogeneous regression problem (c) Heterogeneous classification problem.

FIGURE 3. Distributed Optimization experiments

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.

2.6. 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., 2017b] 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.

In this appendix, we present the missing proofs and details for this Chapter. In particular, Section 2.A is devoted to proving Theorem 11, which shows that nodes are active a constant fraction of the time in average. Then, we compare the speed of ESDACD with that of standard gossip and of SSDA in Section 2.B, and finally prove the detailed convergence rate of ESDACD in Section 2.C.

2.A. Detailed average time per iteration proof

The goal of this section is to prove Theorem 11. 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 3. *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 $i \in \{1,..,n\}$:

$$T_i(t) = \max_{w \in \mathbb{N}, X^t(i,w)>0} w. \tag{2.A.1}$$

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 (2.A.1) 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,

$$T_{\max}(t) = \max_i \max_{w\in\mathbb{N}, X^t(i,w)>0} w = \max_{w\in\mathbb{N}, \sum_i X^t(i,w)>0} w = \sum_{w\in\mathbb{N}} \mathbb{1}\left(\sum_{i=1}^n X^t(i,w) \geq 1\right), \quad (2.A.2)$$

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 4. *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 $\theta > 0$*

$$\sum_{p\geq\theta t} \mathbb{E}\left[X^t(i,p)\right] \leq (1+\delta)^t \, \mathbb{P}\left[Binom(2\delta_2, t) \geq \theta t\right]. \quad (2.A.3)$$

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

$$\mathbb{E}\left[X^{t+1}(i,w)\right] = (1-p_i)\,\mathbb{E}\left[X^t(i,w)\right] + \sum_j p_{ij}\mathbb{E}\left[X^t(j,w-1)\right] + p_i\mathbb{E}\left[X^t(i,w-1)\right]. \quad (2.A.4)$$

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:

$$\bar{X}^{t+1}(w) = (1-\underline{p})\,\bar{X}^t(w) + 2\bar{p}\bar{X}^t(w-1). \quad (2.A.5)$$

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

$$\phi^t(z) = \left(1 - \underline{p} + 2\bar{p}z\right)^t. \quad (2.A.6)$$

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

$$\phi_t(z) \leq (1+\delta)^t \left(1 - 2\delta_2 + \frac{2\delta_2}{1+\delta}z\right)^t \leq (1+\delta)^t \left(1 - 2\delta_2 + 2\delta_2 z\right)^t = (1+\delta)^t \phi_{bin}(2\delta_2, t)(z), \quad (2.A.7)$$

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

$$\lim_{t\in\mathbb{N}} \sum_{w\geq\theta t} \mathbb{E}\left[X^t(i,w)\right] = 0 \quad (2.A.8)$$

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

$$\ln \mathbb{P}\left[Binom(2\delta_2, t) \geq \theta t\right] \leq -tD(\theta||2\delta_2), \tag{2.A.9}$$

where $D(p||q) = p\ln\frac{p}{q} + (1-p)\ln\frac{1-p}{1-q}$ so applying Lemma 4 yields:

$$\sum_{w\geq\theta t} \mathbb{E}\left[X^t(i,w)\right] \leq e^{-t[D(\theta||2\delta_2)-\ln(1+\delta)]}. \tag{2.A.10}$$

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

$$D(\theta||2\delta_2) = 2\delta_2\ln(\frac{2\delta_2}{\theta}) + (1-2\delta_2)\ln\frac{1-2\delta_2}{1-\theta} \geq 2\delta_2\ln(\frac{2\delta_2}{\theta}) - 2\delta_2 + \theta \tag{2.A.11}$$

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:

$$D(\theta||2\delta_2) \geq 2\delta_2\left[2 - \ln(3 + \frac{\delta}{2\delta_2})\right] + \delta > \delta \geq \ln(1+\delta). \tag{2.A.12}$$

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

□

2.B. Execution speed comparisons

2.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 $W_{ij} = Ae_{ij}e_{ij}^TA^T$ 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

$$\theta_{ESDACD} = \min_{ij} \frac{p_{ij}}{\mu_{ij}\sqrt{\sigma_i^{-1} + \sigma_j^{-1}}} \frac{\sqrt{\lambda_{\min}^+(A^TA)}}{\sqrt{e_{ij}A^+Ae_{ij}}} \geq \sqrt{\frac{\lambda_{\min}^+(AA^T)}{cnE}} \tag{2.B.1}$$

where c is a constant independent of the size of the graph coming from Assumption 3.

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

$$\theta_{ESDACD} \geq \frac{c'}{\sqrt{n}}\sqrt{\theta_{\text{gossip}}} \tag{2.B.2}$$

with $c' = c^{-\frac{1}{2}}$. □

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:

$$\theta_{ESDACD} \geq \min_{ij} \frac{p_{ij}}{\mu_{ij}\sqrt{L_{\max}(\sigma_i^{-1}+\sigma_j^{-1})e_{ij}^T A^+ A e_{ij}}} \sqrt{\lambda_{\min}^+(A^TA)}. \tag{2.B.3}$$

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

$$\theta_{ESDACD} \geq p_{\min}\sqrt{\frac{\lambda_{\max}(AA^T)}{\max_{ij} e_{ij}^T A^+ A e_{ij}}}\sqrt{\frac{\gamma}{\kappa}}. \tag{2.B.4}$$

Therefore, combining this with Theorem 11 and Assumption 3 gives:

$$\frac{\theta_{ESDACD}}{\bar{\tau}_{ESDACD}} \geq \frac{p_{\min}\sqrt{E}}{c\bar{p}\tau_{\max}}\sqrt{\frac{\lambda_{\max}(AA^T)}{n}}\sqrt{\frac{\gamma}{\kappa}} \geq \frac{c'}{\tau_{\max}}\frac{p_{\min}}{p_{\max}}\sqrt{\frac{d_{\min}}{d_{\max}}}\sqrt{\frac{E}{nd_{\max}}}\sqrt{\frac{\gamma}{\kappa}} \tag{2.B.5}$$

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 2 to get that there exists c'' such that:

$$\frac{\theta_{ESDACD}}{\bar{\tau}_{ESDACD}} \geq \frac{c''}{\tau_{\max}}\sqrt{\frac{\gamma}{\kappa}} = c''\frac{\theta_{SSDA}}{\bar{\tau}_{SSDA}} \tag{2.B.6}$$

□

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.

2.C. Detailed rate proof

The proof of Theorem 10 is detailed in this section. Recall that we note A^+ the pseudo-inverse of A and we define the scalar product $\langle x, y\rangle_{A^+A} = x^TA^+Ay$. 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 6 (Strong convexity of F_A^*). *For all $x, y \in \mathbb{R}^E$,*

$$F_A^*(x) - F_A^*(y) \geq \nabla F_A^*(y)^T(x-y) + \frac{\sigma_A}{2}\|x-y\|_{A^+A}^2 \tag{2.C.1}$$

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

PROOF. Inequality (2.C.1) 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 $\|Aw\|^2 = \|Aw\|_{A^+A}^2 \geq \lambda_{\min}^+(A^TA)\|w\|_{A^+A}^2$ for $w = x - y$. More specifically:

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

□

LEMMA 7 (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,*

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

Equation (2.C.2) 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 $A e_{ij} = \mu_{ij} (e_i - e_j)$ 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 10. 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:

$$
a_{t+1}^2 S^2 = A_{t+1} B_{t+1} \tag{2.C.3}
$$

$$
B_{t+1} = B_t + \sigma_A a_{t+1} \tag{2.C.4}
$$

$$
A_{t+1} = A_t + a_{t+1} \tag{2.C.5}
$$

$$\alpha_t = \frac{a_{t+1}}{A_{t+1}} \tag{2.C.6}$$

$$\beta_t = \frac{\sigma_A a_{t+1}}{B_{t+1}}. \tag{2.C.7}$$

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

$$y_t = \frac{(1-\alpha_t)x_t + \alpha_t(1-\beta_t)v_t}{1-\alpha_t\beta_t} \tag{2.C.8}$$

$$v_{t+1} = (1-\beta_t)v_t + \beta_t y_t - \frac{a_{t+1}}{B_{t+1}p_{ij}}U_{ij}\nabla F_A^*(y_t) \tag{2.C.9}$$

$$x_{t+1} = y_t - \frac{1}{\mu_{ij}^2(\sigma_i^{-1}+\sigma_j^{-1})}U_{ij}\nabla F_A^*(y_t). \tag{2.C.10}$$

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 $g_{ij}(y_t) = \frac{a_{t+1}}{B_{t+1}p_{ij}}U_{ij}\nabla F_A^*(y_t)$.

$$\|v_{t+1}-x^*\|_{A^+A}^2 = \|w_t-x^*\|_{A^+A}^2 + \|\frac{a_{t+1}}{B_{t+1}p_{ij}}U_{ij}\nabla F_A^*(y_t)\|_{A^+A}^2 - 2\frac{a_{t+1}}{B_{t+1}p_{ij}}\nabla F_A^*(y_t)^T U_{ij}A^+A(w_t-x^*). \tag{2.C.11}$$

Then,

$$\mathbb{E}_{ij}[\frac{a_{t+1}}{B_{t+1}p_{ij}}\nabla F_A^*(y_t)^T U_{ij}] = \sum_{ij} p_{ij}\frac{a_{t+1}}{B_{t+1}p_{ij}}\nabla F_A^*(y_t)^T U_{ij} = \frac{a_{t+1}}{B_{t+1}}\nabla F_A^*(y_t)^T. \tag{2.C.12}$$

Therefore, Equation (2.C.11) can be rewritten:

$$\mathbb{E}\left[r_{t+1}^2\right] \leq \mathbb{E}\left[\|w_t - x^*\|_{A^+A}^2\right] + \mathbb{E}\left[\frac{e_{ij}^T A^+ A e_{ij} a_{t+1}^2}{B_{t+1}^2 p_{ij}^2}\|U_{ij}\nabla F_A^*(y_t)\|^2\right] - 2\frac{a_{t+1}}{B_{t+1}}\nabla F_A^*(y_t)^T(w_t - x^*). \tag{2.C.13}$$

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 (2.C.2) in Lemma 7.

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:

$$\|v_{t+1}-x^*\|_{A^+A}^2 \leq \|w_t-x^*\|_{A^+A}^2 + \frac{2S^2a_{t+1}^2}{B_{t+1}^2}\left[F_A^*(y_t) - \mathbb{E}\left[F_A^*(x_{t+1})\right]\right] - 2\frac{a_{t+1}}{B_{t+1}}\nabla F_A^*(y_t)^T(w_t-x^*). \tag{2.C.14}$$

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:

$$B_{t+1}r_{t+1}^2 \leq B_t r_t^2 + \beta_t B_{t+1}\|y_t - x^*\|_{A^+A}^2 + \frac{2S^2a_{t+1}^2}{B_{t+1}}\left[F_A^*(y_t) - \mathbb{E}\left[F_A^*(x_{t+1})\right]\right] - 2a_{t+1}\nabla F_A^*(y_t)^T(w_t - x^*). \tag{2.C.15}$$

We can now use Equation (2.C.1) of Lemma 6 (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 (2.C.15) and we use the fact that $B_{t+1}\beta_t = a_{t+1}\sigma_A$ so that:

$$
B_{t+1}r_{t+1}^2 \leq B_t r_t^2 + 2A_{t+1}\left[F_A^*(y_t) - \mathbb{E}\left[F_A^*(x_{t+1})\right]\right] - 2\left[(A_{t+1} - A_t)F_A^*(x^*) - A_{t+1}F_A^*(y_t) + A_t F_A^*(x_t)\right], \tag{2.C.16}
$$

and so:

$$
B_{t+1}r_{t+1}^2 - B_t r_t^2 \leq 2A_t\left[F_A^*(x_t) - F_A^*(x^*)\right] - 2A_{t+1}\left[\mathbb{E}\left[F_A^*(x_{t+1})\right] - F_A^*(x^*)\right]. \tag{2.C.17}
$$

By summing over all inequalities, we get that

$$
2A_t\mathbb{E}\left[F_A^*(x_t) - F_A^*(x^*)\right] + B_t\mathbb{E}[r_t^2] \leq r_0^2. \tag{2.C.18}
$$

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 (2.C.5) and Equation (2.C.6) to obtain

$$
a_{t+1}(\alpha_t^{-1} - 1) = A_t \tag{2.C.19}
$$

$$
a_{t+1}(\beta_t^{-1} - 1) = \frac{B_t}{\sigma_A} \tag{2.C.20}
$$

For $t = 0$, we can combine equations (2.C.19) and (2.C.20) 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 (2.C.19) 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 (2.C.20) in the same way to prove that $B_{t+1} = \sigma_A A_{t+1}$.

Then, we use equations (2.C.19) and (2.C.20) 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 (2.C.3), 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 (2.C.10) and (2.C.9). 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. □

CHAPTER 3

Accelerated Variance-reduced decentralized stochastic optimization

In this chapter, we introduce a lower bound for decentralized stochastic optimization, and match it with an accelerated dual method, based on an augmented graph formulation that combines the strength of APCG [Lin et al., 2015b] and ESDACD, presented in Chapter 2. More specifically, we introduce an efficient **A**ccelerated **D**ecentralized stochastic algorithm for **F**inite **S**ums named ADFS, which uses local stochastic proximal updates (which are generally more expensive than gradient updates) and decentralized communications between nodes. On n machines, ADFS minimizes the objective function with nm samples in the same time it takes optimal algorithms to optimize from m samples on one machine. This scaling holds until a critical network size is reached, which depends on communication delays, on the number of samples m, and on the network topology. We give a lower bound of complexity to show that ADFS is optimal among decentralized algorithms. To derive ADFS, we first develop an extension of the accelerated proximal coordinate gradient algorithm to arbitrary sampling. Then, we apply this coordinate descent algorithm to a well-chosen dual problem based on an augmented graph approach, leading to the general ADFS algorithm. We illustrate the improvement of ADFS over state-of-the-art decentralized approaches with experiments.

This chapter is based on the paper *An Optimal Algorithm for Decentralized Finite Sum Optimization* [Hendrikx, Bach, and Massoulie, 2020b], accepted for publication at the SIAM Journal on Optimization (SIOPT), which is itself based on the paper *An Accelerated Decentralized Stochastic Proximal Algorithm for Finite Sums* [Hendrikx et al., 2019b], published at NeurIPS 2019. Yet, these articles have been reworked, and in particular the randomized gossip aspect from the conference paper has been reintegrated in Section 3.7.1.

3.1. Introduction

The success of machine learning models is mainly due to their capacity to train on huge amounts of data. Distributed systems can be used to process more data than one computer can store or to increase the pace at which models are trained by splitting the work among many computing nodes. In this work, we focus on problems of the form:

$$\min_{\theta\in\mathbb{R}^d} \sum_{i=1}^{n} f_i(\theta), \qquad \text{where} \qquad f_i(\theta) = \sum_{j=1}^{m} f_{ij}(\theta) + \frac{\sigma_i}{2}\|\theta\|^2. \tag{3.1.1}$$

This is the typical ℓ_2-regularized empirical risk minimization problem with n computing nodes that have m local training examples each. The function f_{ij} represents the loss function for the j-th training example of node i and is assumed to be convex and L_{ij}-smooth [Nesterov, 2013c, Bubeck, 2015]. This kind of problems arises in machine learning problems such as

image recognition [Goyal et al., 2017], as well as in other applications such as distributed resource allocation [Xiao and Boyd, 2006] or distributed power control [Molzahn et al., 2017].

These problems are usually solved by first-order methods, and the basic distributed algorithms compute gradients in parallel over several machines [Nedic and Ozdaglar, 2009]. Another way to speed up training is to use *stochastic* algorithms [Bottou, 2010, Defazio et al., 2014a, Johnson and Zhang, 2013b], that take advantage of the finite sum structure of the problem to use cheaper iterations while preserving fast convergence. Lower bounds with matching optimal algorithms exist separately in both the finite-sum [Lan and Zhou, 2017] and the distributed setting [Scaman et al., 2017b]. This paper aims at bridging the gap between these two lines of work when local functions are smooth and strongly convex. In particular, we give lower complexity bounds for the distributed finite-sum setting, as well as ADFS, an algorithm that matches these bounds. Our contributions are the following, ordered by appearance in the paper:

(1) Tight lower complexity bounds. We recover as special cases the bounds from Scaman et al. [2017b] when $m = 1$ (local functions are not finite sums), and the bounds from Lan and Zhou [2017] when $n = 1$ (there is only one machine).
(2) Generalization of Accelerated Proximal Coordinate Gradient [Lin et al., 2015b, Fercoq and Richtárik, 2015] to arbitrary sampling of blocks and strong convexity in a subspace.
(3) ADFS, a decentralized stochastic algorithm that matches our lower complexity bounds, and recovers the rate of MSDA [Scaman et al., 2017b] when $m = 1$ and the rate of optimal single machine stochastic algorithms [Lin et al., 2015b, Defazio, 2016] when $n = 1$.

The present paper is an extended journal version of the conference paper that introduces ADFS [Hendrikx et al., 2019b]. In particular, this paper presents a more flexible version of ADFS that can use synchronous rounds instead of local operations with global scheduling, and the contributions listed above were not present in the original work. We discuss the differences with the conference paper more in details later in the article. We now precisely define our setting, and discuss relevant related work.

3.2. Model and notations

3.2.1. Optimization problem. In the rest of this paper, following Scaman et al. [2017b], we assume that:

- Each node (computing unit) $i \in \{1, ..., n\}$ can compute first-order characteristics, such as the gradient of its own functions ∇f_{ij}, the gradient of the Fenchel conjugate of its local function ∇f_i^*, or the proximal operator

$$\text{prox}_{\eta f_{ij}}(x) = \arg\min_v \frac{1}{2\eta}\|v - x\|^2 + f_{ij}(v) \quad \text{for } x \in \mathbb{R}^d. \tag{3.2.1}$$

  We assume that computing first-order characteristics for one function f_{ij} takes time 1, and so that computing them for the function f_i takes time m. This hides the fact that computing the proximal operator of a function (which is required by ADFS) is generally significantly more expensive than computing its gradient. Yet, this enables easier comparison between methods, and the difference between computing the proximal operator compared to the gradient of a single function is only of a

constant factor in the case of generalized linear models such as least-squares or logistic regression.

- Nodes are linked by a communication network and can only exchange messages (*i.e.*, vectors in $\mathbb{R}^d$) with their neighbours. We assume that communications take time τ. There are at this point no other restrictions on the communications, that can happen asynchronously and in parallel.

Following notations from Xiao et al. [2019b], we define the batch condition number κ_b, which is a classical quantity in optimization, such that for all i,

$$\kappa_b \geq L_b(i)/\sigma_i \text{ where } \forall x,\ L_b(i) \geq \lambda_{\max}(\nabla^2 f_i(x)). \tag{3.2.2}$$

Similarly, we define the stochastic condition number κ_s, which is a classical quantity in the analysis of finite sum optimization problems, such that

$$\kappa_s \geq 1 + \frac{1}{\sigma_i}\sum_{j=1}^m L_{ij} \text{ where } \forall x,\ L_{ij} \geq \lambda_{\max}(\nabla^2 f_{ij}(x)). \tag{3.2.3}$$

Batch optimization methods such as gradient descent have an iteration complexity which is proportional to κ_b, but they need to evaluate a full gradient (*i.e.* m individual gradients) at each step. On the other hand, the iteration complexity of stochastic variance-reduced algorithms [Johnson and Zhang, 2013b, Defazio et al., 2014a, Shalev-Shwartz and Zhang, 2013] depends on $m + \kappa_s$, but only use one individual gradient at each iteration. Therefore, stochastic variance-reduced methods improve over batch methods by replacing their $O(m\kappa_b)$ time complexity by $O(m+\kappa_s)$. Yet, since $\nabla^2 f_i(x) = \sigma_i I_d + \sum_{j=1}^m \nabla^2 f_{ij}(x) \preccurlyeq (\sigma_i + \sum_{j=1}^m L_{ij}) I_d$ then we directly obtain that $L_b(i) \leq \sigma_i + \sum_{i=1}^m L_{ij}$, and so $\kappa_b \leq \kappa_s$. Similarly, $\nabla^2 f_i(x) \succcurlyeq \nabla^2 f_{ij}(x)$ and so $L_{ij} \leq L_b(i)$ for all j. Therefore, we always have

$$(m+1)\kappa_b \geq \kappa_s \geq \kappa_b. \tag{3.2.4}$$

This means that finite-sum methods gain nothing in the worst case, in which all f_{ij} are independent. However, the practical superiority of these methods suggests that $\kappa_s << m\kappa_b$ in many applications since samples are often correlated.

The upper bound on κ_s is tight when $\max_x \lambda_{\max}(\nabla^2 f_i(x)) = \max_x \lambda_{\max}(\nabla^2 f_{ij}(x))$. Equality happens in particular in the extreme case in which all $\nabla^2 f_{ij}(x)$ are orthogonal, meaning that the sum is separable and optimization can be performed separately for each function. Considering least squares regression problems is also convenient to understand the difference between κ_s and κ_b more in details. In particular, if we denote by $C \in \mathbb{R}^{m\times m}$ the covariance matrix of the data, then $\kappa_b = \lambda_{\max}(C)$ the largest eigenvalue of C and $\kappa_s = \mathrm{Tr}(C)$. In this case, it is clear that $\kappa_s << m\kappa_b$ unless the covariance matrix is close to isotropic. In summary, our goal is to replace the $m\kappa_b$ computational time factors by $m + \kappa_s$. Whether this improves the global time complexity depends on the structure of the problem (and thus of the data) but this is generally verified in practice. Note that a specific sampling scheme is required to obtain complexities that depend on κ_s as an average of local smoothnesses instead of a max, which is why arbitary sampling (with minibatches) versions of existing stochastic algorithms are sometimes analyzed [Qian et al., 2019b,a, Sebbouh et al., 2019].

3.2.2. Decentralized Communications. The focus of this paper is on the *decentralized* setting. In this case, *gossip* algorithms [Boyd et al., 2006, Nedic and Ozdaglar, 2009, Shi et al., 2015a, Nedic et al., 2017] are generally used. Gossip communication steps consist

Algorithm	Synchrony	Stochastic	Time
Point-SAGA [Defazio, 2016]	N/A	✓	$nm + \sqrt{nm\kappa_s}$
MSDA [Scaman et al., 2017b]	Global	×	$\sqrt{\kappa_b}\left(m + \frac{\tau}{\sqrt{\gamma}}\right)$
ESDACD [Hendrikx et al., 2019a]	Local	×	$(m+\tau)\sqrt{\frac{\kappa_b}{\gamma}}$
DSBA [Shen et al., 2018]	Global	✓	$\left(m + \kappa_s + \gamma^{-1}\right)(1+\tau)$
ADFS-Asynch [Hendrikx et al., 2019b]	Local	✓	$m + \sqrt{m\kappa_s} + (1+\tau)\sqrt{\frac{\kappa_s}{\gamma}}$
ADFS-Synch (This paper)	Global	✓	$m + \sqrt{m\kappa_s} + \tau\sqrt{\frac{\kappa_{\text{comm}}}{\gamma}}$

Table 1. Comparison of various state-of-the-art decentralized algorithms to reach accuracy ε in regular graphs. Constant factors are omitted, as well as the $\log(\varepsilon^{-1})$ factor in the Time column. The reported runtime for Point-SAGA corresponds to running it on a single machine with nm samples. To allow for direct comparison, we assume that computing a dual gradient of a function f_i as required by MSDA and ESDACD takes time m, although it is generally more expensive than to compute m separate proximal operators of single f_{ij} functions. Rates reported are for a homogeneous setting, *i.e.*, when all nodes have the same strong convexity parameter. For generalized linear models such as logistic regression, the κ_{comm} term in the rate of ADFS-Synch is defined in Lemma 11 and is of order $\kappa_{\text{comm}} = O(\kappa_b)$.

in averaging gradients or parameters with neighbours, and can thus be abstracted as multiplication by a so-called gossip matrix W, which is an $n \times n$ symmetric positive semi-definite matrix such that $\text{Ker}(W) = \text{Span}(\mathbb{1})$ where $\mathbb{1}$ is the constant vector of all ones, and $\text{Span}(\mathbb{1})$ denotes the vector space spanned by this vector. Besides, W is defined on the edges of the network, meaning that $W_{k\ell} = 0$ if $\ell \neq k$ and $\ell \notin \mathcal{N}(k)$, the set of the neighbours of node k.

A simple choice of gossip matrix is L, the Laplacian matrix of the graph, which is such that $L_{k\ell} = \text{degree}(k)$ if $k = \ell$, $L_{k\ell} = -1$ if $k \in \mathcal{N}(\ell)$ and $L_{k\ell} = 0$ otherwise. We denote $\lambda^+_{\min}(W)$ the smallest non-zero eigenvalue of the matrix W, and the *eigengap* of the gossip matrix (also called spectral gap) is defined as $\gamma = \lambda^+_{\min}(W)/\lambda_{\max}(W)$. This natural constant appears in the running time of many decentralized algorithms, and $\gamma^{-1/2}$ is often close to the diameter of the graph. For instance, $\gamma^{-1/2} = 1$ for the complete graph, $\gamma^{-1/2} = 2n/\pi$ for linear graphs and $\gamma^{-1/2} = O(\sqrt{n})$ for the 2D grid. More generally, $\gamma^{-1/2} \geq \frac{\Delta}{2\sqrt{2}\ln_2(n)}$ for regular networks [Alon and Milman, 1985], and there is a simple distributed choice of weights (the so-called Metropolis weights), such that $\gamma^{-1/2} = O(n)$ for any graph (see, *e.g.*, Olshevsky [2017]).

3.3. Related work

The next paragraphs discuss the state of the art for both distributed and stochastic methods, and Table 1 sums up the speeds of the main decentralized algorithms for solving Problem (3.1.1). Point-SAGA [Defazio, 2016], an optimal single-machine algorithm, is also presented for comparison.

Centralized gradient methods. A simple way to split work between nodes is to distribute gradient computations and to aggregate them on a parameter server. Provided the network is fast enough, this allows the system to learn from the datasets of n workers in the same time one worker would need to learn from its own dataset. Yet, these approaches are very sensitive to stochastic delays, slow nodes, and communication bottlenecks. Asynchronous methods may be used [Recht et al., 2011, Leblond et al., 2017, Xiao et al., 2019b] to address the first two issues, but computing gradients on older (or even inconsistent) versions of the parameter harms convergence [Chen et al., 2016]. Therefore, this paper focuses on decentralized algorithms, which are generally less sensitive to communication bottlenecks [Lian et al., 2017a].

Decentralized gradient methods. In their synchronous versions, decentralized algorithms alternate rounds of computations (in which all nodes compute gradients with respect to their local data) and communications, in which nodes exchange information with their direct neighbors. Typical examples include dual averaging [Duchi et al., 2012b], EXTRA [Shi et al., 2015a], DIGing [Nedic et al., 2017] or NIDS [Li et al., 2019]. The convergence theory of these algorithms has been refined over time so that EXTRA and NIDS are now known to require time $O((\kappa_b + \gamma^{-1})(m + \tau))\log(\varepsilon^{-1}))$ to reach precision ε [Jakovetić, 2018, Xu et al., 2020c, Li and Lin, 2020]. Chebyshev and Catalyst acceleration [Lin et al., 2015a] can then be used to obtain the (batch) optimal $O(\sqrt{\kappa_b}(1 + \tau/\sqrt{\gamma})\log(\varepsilon^{-1}))$ rate up to log factors [Li and Lin, 2020]. Decentralized algorithms can also be obtained through the penalty method [Li et al., 2018, Dvinskikh and Gasnikov, 2019], which consists in applying standard optimization algorithms to problems augmented with a well-chosen Laplacian penalty. Yet, these algorithms converge to the solution of the penalized problem instead of the original one. Dual approaches [Scaman et al., 2017b, Uribe et al., 2020] are also successful to build decentralized algorithms, and in particular MSDA [Scaman et al., 2017b] is optimal with respect to the constants γ and κ_b, among the class of batch synchronous algorithms. Yet, dual approaches generally assume access to the proximal operator or the gradient of the Fenchel conjugate of the local functions, which is not very practical in general since it requires solving a subproblem at each step. Recently, APAPC Kovalev et al. [2020] obtained the same batch optimal rate, but using primal gradients only. OPTRA [Xu et al., 2020a], another primal-dual algorithm, achieves similar optimal results in the convex (not strongly) regime. The algorithms mentioned above are primarily designed for smooth and (strongly) convex problems. For non-convex problems, one can use NEXT [Di Lorenzo and Scutari, 2016], which is based on the idea of tracking the global gradient, and which was later extended as SONATA, which handles time-varying digraphs [Sun et al., 2016]. SONATA also enjoys fast convergence rates in the strongly convex setting, and even faster rates when local objectives are statistically similar [Sun et al., 2019b], just like a decentralized variant of DANE [Shamir et al., 2014, Li et al., 2020a]. Note that, similarly to PG-EXTRA [Shi et al., 2015b], P2D2 [Alghunaim et al., 2019] or NIDS [Li et al., 2019], NEXT and SONATA can also handle composite problems, in which an extra convex non-smooth regularizer is added to the objective, and for which Xu et al. [2020c] provides a general unified analysis framework.

Instead of performing global synchronous updates, some approaches inspired from gossip algorithms [Boyd et al., 2006] use randomized pairwise communications [Nedic and Ozdaglar, 2009, Johansson et al., 2009, Colin et al., 2016]. This for instance allows fast nodes to

perform more updates in order to benefit from their increased computing power. These randomized algorithms generally do not suffer from the usual worst-case analyses of bounded-delay asynchronous algorithms, and can thus have fast rates because the step-size does not need to be reduced in the presence of delays. Such algorithms include Iutzeler et al. [2013], Bianchi et al. [2015], Koloskova et al. [2020], Notarnicola and Notarstefano [2016] or Cannelli et al. [2020], which also deals with delayed gradients. ESDACD [Hendrikx et al., 2019a], which is very related to the conference version of ADFS [Hendrikx et al., 2019b], is a dual pairwise gossip algorithm that achieves the same optimal speed as MSDA when batch computations are faster than communications ($\tau > m$). We refer the interested reader to Assran et al. [2020] for a survey on asynchronous optimization.

Stochastic algorithms for finite sums. So far, we have mainly presented *batch* methods that compute *full* gradient steps of each function f_i. Stochastic methods perform updates based on randomly chosen functions f_{ij}. In the smooth and strongly convex setting, they can be coupled with *variance reduction* [Schmidt et al., 2017, Shalev-Shwartz and Zhang, 2013, Johnson and Zhang, 2013b, Defazio et al., 2014a] and *acceleration*, to achieve the $m + \sqrt{m\kappa_s}$ optimal finite-sum rate, which significantly improves over the $m\sqrt{\kappa_b}$ batch optimum when the dataset is large. Examples of such methods include Accelerated-SDCA [Shalev-Shwartz and Zhang, 2014], APCG [Lin et al., 2015b], Point-SAGA [Defazio, 2016] or Katyusha [Allen-Zhu, 2017].

Decentralized stochastic methods. The simplest (yet efficient) stochastic decentralized algorithms are decentralized versions of the parallel SGD algorithm [Lian et al., 2017a, Tang et al., 2018b, Koloskova et al., 2019a, Nadiradze et al., 2019, Assran et al., 2019, Koloskova et al., 2020]. Conversely, [Zhang and You, 2019, Pu and Nedić, 2020] develop stochastic variants of the celebrated gradient tracking approach (at the heart of NEXT or DIGing, among others), in which each node estimates the global gradient. Yet, these methods only converge to a neighbourhood of the solution when using a constant step-size. In the smooth and strongly convex setting, DSA [Mokhtari and Ribeiro, 2016] and later DSBA [Shen et al., 2018] are two linearly converging stochastic decentralized algorithms. DSBA uses the proximal operator of individual functions f_{ij} to significantly improve over DSA in terms of rates. Yet, DSBA does not enjoy the $\sqrt{m\kappa_s}$ accelerated rate, and needs an excellent network with very fast communications. Indeed, nodes need to communicate each time they process a single sample, resulting in many communication steps. Other recent variance-reduced methods include GT-SAGA/SVRG [Xin et al., 2020c], but the rates of these methods depend on κ_s^2 instead of κ_s, and are thus much slower (although only gradients of local functions are used instead of prox). Li et al. [2020a] also consider the same setting, and prove super fast convergence of a network SVRG algorithm under the assumption that all local functions are very similar. Therefore, to the best of our knowledge, there is no decentralized stochastic algorithm with accelerated linear convergence rate or low communication complexity without sparsity assumptions (*i.e.*, sparse features in linear supervised learning).

ADFS. This paper main contribution is a locally synchronous **A**ccelerated **D**ecentralized stochastic algorithm for **F**inite **S**ums, named ADFS. It reduces to APCG for empirical risk minimization [Lin et al., 2015b] in the limit case $n = 1$ (single machine), and therefore then has a $m + \sqrt{m\kappa_s}$ convergence rate. Besides, this rate stays unchanged when the number of machines grows, meaning that ADFS can process n times more data than APCG in the same

amount of time on a network of size n. This scaling lasts as long as $\tau\sqrt{\kappa_{\text{comm}}}\gamma^{-\frac{1}{2}} < m+\sqrt{m\kappa_s}$, meaning that the number of nodes can be arbitrarily large as long as delays are small enough. Therefore, ADFS outperforms both MSDA and DSBA, combining optimal network scaling with the efficient distribution of optimal sequential finite-sum algorithms. Note however that, similarly to DSBA and Point-SAGA, ADFS requires evaluating $\text{prox}_{f_{ij}}$, which requires solving a local optimization problem. Yet, in the case of linear models such as logistic regression, it is only a constant factor slower than computing ∇f_{ij}, and it is especially much faster than computing the gradient of the conjugate of the full dual functions ∇f_i^* required by ESDACD and MSDA.

Follow-up works. While this work was under review, accelerated variance reduced versions of EXTRA and DIGing were developed in a preprint [Li et al., 2020c]. Accelerated VR EXTRA obtains optimal rates using primal gradients only. Yet, it leverages SVRG-style [Johnson and Zhang, 2013b] variance reduction, and thus needs to periodically recompute full gradients of the local objectives. Besides, in order to obtain optimal rates, the computation-communication ratio needs to be balanced using minibatches of size b, whereas the probability to compute and communicate can be tuned directly using ADFS. Finally, only fixed synchronous undirected graphs are considered, whereas ADFS also handles randomized pairwise communications. We thus believe that ADFS is still a relevant method for accelerated decentralized stochastic variance-reduced optimization, especially when dealing with linear models.

Improvements over the conference paper. This paper is based on the ADFS conference paper [Hendrikx et al., 2019b]. Yet, it is not a strict extension, and some parts have been removed in order to ease the reading. In particular, the locally synchronous aspect of ADFS has been dropped in favor of standard synchronous gossip, which allows to remove the sections about time and scheduling. This paper is based on arguments that are similar to the ones used in the conference paper, but it presents new and stronger results. First of all, we introduce a lower bound that was not present in the conference paper. Then, we extend the accelerated proximal coordinate descent algorithm, which is the algorithmic core of ADFS, to work with blocks of coordinates. This allows to present a synchronous version of ADFS, which is both simpler and faster when communication and computation delays are homogeneous (the same for all nodes and edges). Furthermore, we introduce the constant κ_{comm}, which captures the impact of the relationship between the topology of the graph and the regularity of local functions on the iteration complexity of ADFS. This allows us to obtain tight results on the communication complexity of ADFS and show that ADFS is actually optimal since it matches the lower bound. Note that the locally synchronous version of ADFS from the conference paper did not enjoy optimal runtime because of scheduling issues and a looser analysis. Therefore, and although they build on the same ideas as the conference paper, all results presented in this paper are novel and contribute to building a much more consistent theory. In this thesis, the contributions from the conference paper have been reintegrated in Section 3.7.1.

The first contribution of this paper is a lower bound for distributed finite sum optimization, presented in Section 3.4. Then, we introduce in Section 3.5 our second contribution, a generalization of APCG that works with arbitrary sampling of blocks of coordinates. Our last contribution is ADFS, obtained by applying the previous APCG algorithm to a novel

augmented graph approach formulation presented in Section 3.6.1. The generic ADFS algorithm is presented in Section 3.6.2. Finally, Section 3.6.5 presents a relevant choice of parameters leading to the rates shown in Table 1, and an experimental comparison is done in Section 3.8.

3.4. Optimal rates

Many of the algorithms discussed in the previous sections are proven to be optimal in specific settings. In particular, APCG (when applied to the dual of empirical risk minimization problems) and Point-SAGA are proven to be optimal among single-machine algorithms to solve finite-sum problems [Lan and Zhou, 2017]. Similarly, MSDA is optimal among batch decentralized algorithms [Scaman et al., 2017b]. Although other optimality results have recently been proven when removing the strong convexity and smoothness assumptions in the distributed setting [Scaman et al., 2018], there is, to the best of our knowledge, no lower bound for distributed optimization when local functions are themselves finite sums. We fill this gap in this section by extending the decentralized lower bound of Scaman et al. [2017b] to the finite sum setting, using worst-case functions inspired from the single-machine finite-sum lower bound Lan and Zhou [2017].

3.4.1. Black Box Model. The notion of black-box optimization procedure that we use is largely based on Scaman et al. [2018]. The main difference is that nodes have many local functions but they only choose one (possibly at random) at each step to perform their update. More specifically, we consider distributed algorithms that respect:

(1) **Local memory:** each node i can store past values in an internal memory $\mathcal{M}_{i,t} \subset \mathbb{R}^d$ at time $t \geq 0$. The values in this local memory can come either from local computation or communication, so that for all $i \in \{1, \cdots n\}$, $\mathcal{M}_{i,t} \subset \mathcal{M}_{i,t}^{\text{comm}} \cup \mathcal{M}_{i,t}^{\text{comp}}$.

(2) **Local computation:** each node can, at time t, compute $\nabla f_{i,\zeta_t}(\theta)$, $\nabla f_{i,\zeta_t}^*(\theta)$ and $\text{prox}_{\eta f_{i,\zeta_t}}(\theta)$ for some $\eta > 0$, where $\zeta_t \in \{1, \cdots, m\}$ is fixed for a given t (but may be chosen by the algorithm). This means that

$$\mathcal{M}_{i,t}^{\text{comp}} = \text{Span}\left(\left\{\theta, \nabla f_{i,\zeta_t}(\theta), \nabla f_{i,\zeta_t}^*(\theta), \text{prox}_{\eta f_{i,\zeta_t}}(\theta) : \theta \in \mathcal{M}_{i,t-1}\right\}, \eta \geq 0\right).$$

(3) **Local communication:** each node can, at time t, share a value to its neighbours so that for all $i \in \{1, \cdots, n\}$, $\mathcal{M}_{i,t}^{\text{comm}} = \text{Span}\left(\cup_{j \in \mathcal{N}(i)} \mathcal{M}_{j,t-\tau}\right)$.

(4) **Output value:** each node i must, at time specify one vector in its memory as local output of the algorithm, that is, for all $i \in \{1, \cdots, n\}$, $\theta_{i,t} \in \mathcal{M}_{i,t}$.

The main difference with the definition from Scaman et al. [2018] is that at each step, the first order characteristics are only computed for one summand (the one with index ζ_t) of the local finite sum of node i.

3.4.2. Lower bounds. We first present a general lower bound for distributed optimization. More specifically, we show that for any black-box optimization procedure, at least $\Omega((m + \sqrt{m\kappa_s})\log(1/\varepsilon))$ computation steps and $\Omega(\tau\sqrt{\kappa_b/\gamma}\log(1/\varepsilon))$ communication steps are needed. This lower bound is not surprising since it is similar to that of Scaman et al. [2017b], but the lower bound on the computation cost is replaced by the standard finite-sum

lower-bound for the computation cost [Lan and Zhou, 2017]. Theorem 12 shows that the lower bound for both communications and computations can be achieved by the same function. Lower bound proofs for first-order methods usually rely on the fact that in the work case, the algorithms can make progress in at most one dimension per oracle call [Nesterov, 2013c]. This means that the lower bounds are valid only for a number of iterations t that depends on the dimension of the problem. In order to avoid this dependency, we prove a result in ℓ_2, the space of square summable sequences. Yet, a similar result with a similar proof would hold in $\mathbb{R}^d$.

THEOREM 12. *Let $\mathcal{G}$ be a graph of size $n > 0$ and diameter Δ, and $\kappa_b > 0$. There exist $n \times m$ functions $f_{ij} : \ell_2 \to \mathbb{R}$ such that each f_{ij} is convex and L_{ij}-smooth, $f_i \hat{=} \sum_{j=1}^m f_{ij}$ is L_i-smooth and σ_i-strongly convex with L_{ij} and σ_i such that $\kappa_b \geq L_i/\sigma_i \geq L_{ij}/\sigma_i$ for all i, j, and such that if f_i is the local function of node i then for any $t \geq 0$ and black-box procedure that generates and output θ^t such that $(\theta^t)_i \in \ell_2$ is the output value of node i at time t, one has:*

$$2\mathbb{E}[\frac{\|\theta^t - \theta^*\|^2}{\|\theta^0 - \theta^*\|^2}] \geq \left(1 - \frac{2m}{m + \sqrt{m\kappa_s/3)}}\right)^{\frac{4\lceil t \rceil}{m}} + \left(1 - \frac{2}{1 + \sqrt{\kappa_b/3}}\right)^{2 + \frac{2\lceil t \rceil}{\Delta \tau}}.$$

where $\kappa_s \geq \sum_{j=1}^m L_{ij}/\sigma_i$, $q = \frac{\sqrt{\kappa_b/3}-1}{\sqrt{\kappa_b/3}+1}$, and $\theta^* = \arg\min_\theta \sum_{i=1}^n f_i(\theta)$.

PROOF. The proof relies on choosing particular functions that are hard to optimize locally and that require communication. Hard functions f_i are chosen similar to that of Scaman et al. [2017b], so that only a small set a of nodes can actually make progress towards the optimum at a given point in time, meaning that parallelism is very restricted. Then, f_{ij} are chosen such that they only depend on θ_j for $\theta \in (\ell_2)^m$, so that progress along one $j \in \{1, \cdots, m\}$ does not result in progress along the other dimensions, as in Lan and Zhou [2017]. The result is stated with ℓ_2 instead of $(\ell_2)^m$ because if m is finite and $x \in (\ell_2)^m$, then if θ is such that $\theta_{km+i} = (x_i)_k$ for $i \in \{1, \cdots, m\}$ and $k \in \mathbb{N}$ then $\theta \in \ell_2$.

More specifically, we consider Q a set of nodes and Q^c_Δ the set of nodes at distance at least Δ from Q in the graph $\mathcal{G}$. Let $L, \sigma > 0$ be such that $L \geq \sigma$ and $\kappa_b > 3L/\sigma$. Then, we define for $y \in \ell_2$ functions ψ_i^Q such that:

$$\psi_i^Q(y) = \frac{1}{2|Q|}\left[\frac{\sigma}{3}\|y\|^2 + \frac{L-\sigma}{4}(y^\top M_1 y - 2e_1^\top y)\right] \text{ if } i \in Q, \tag{3.4.1}$$

$$\psi_i^Q(y) = \frac{1}{2|Q^c_\Delta|}\left[\frac{\sigma}{3}\|y\|^2 + \frac{L-\sigma}{4} y^\top M_2 y\right] \text{ if } i \in Q^c_\Delta, \tag{3.4.2}$$

$$\psi_i^Q(y) = \frac{\sigma}{6(n - |Q^c_\Delta| - |Q|)}\|y\|^2 \text{ otherwise}, \tag{3.4.3}$$

where M_1 is the infinite block diagonal matrices with $\begin{pmatrix} 1 & -1 \\ -1 & 1 \end{pmatrix}$ on the diagonal and $M_2 = \text{Diag}(1, M_1)$. We then define for all $i \in \{1, \cdots, n\}$ the local subfunctions as:

$$f_{ij} : \theta \in \ell_2^m \mapsto \psi_i^Q(\theta_j). \tag{3.4.4}$$

Note that the regularity conditions on the f_{ij} are respected since $0 \preceq M_1 + M_2 \preceq 4I$. Besides, the solution of $\min_{y \in \ell_2} \sum_{i=1}^n \psi_i^Q(y)$ is y^* such that for $k \geq 1$, the k-th coordinate of y^* is

$y^*(k) = q^k$. We now consider a sequence $\theta^t \in \ell_2^{n\times m}$ generated by a black-box optimization procedure as defined in Section 3.4.1 and such that $\theta^0 = 0$ without loss of generality. In this case, $\theta^* \in \ell_2^{n\times m}$ is such that $\theta^*_{ij} = y^*$, and:

$$\mathbb{E}[\sum_{i=1}^{n}\sum_{j=1}^{m}\frac{\|\theta^t_{ij}-\theta^*_{ij}\|^2}{\|\theta^0_{ij}-\theta^*_{ij}\|^2}] \geq \mathbb{E}[\sum_{i=1}^{n}\sum_{j=1}^{m}\sum_{l\geq k_j(t)}\frac{|\theta^*_{ij}(l)|^2}{\|\theta^0_{ij}-\theta^*_{ij}\|^2}] = n\sum_{j=1}^{m}\mathbb{E}[q^{2k_j(t)}],$$

where $k_j(t)$ is the first index such that $\theta^t_{ij}(l) = 0$ for all i and $l \geq k_j(t)$. Thus, an upper bound on $k_j(t)$ gives a lower bound on the expected error, and we consider two extreme regimes.

Case 1 : Communication bottleneck. The first one consists in considering that computations are instant so that nodes in Q and Q^c_Δ perform a gradient update using function i as soon as they receive a message. Due to the form of the f_{ij}, nodes in Q can only increase $k_j(t)$ if it is odd, and node in Q^c_Δ can only increase $k_j(t)$ if it is even. Considering that a message takes time at least $\Delta\tau$ (with Δ the diameter of the network) to go from Q to Q^c_Δ we get:

$$k_j(t) \leq 1 + \frac{t}{\Delta\tau}, \text{ and so } \mathbb{E}[q^{2k_j(t)}] \geq q^{2+\frac{2t}{\Delta\tau}}. \tag{3.4.5}$$

In this case, the stochastic gradient aspect does not matter and the time taken by the algorithm is lower bounded by the time required for the information to go back and forth between the nodes that can actually make progress.

Case 2 : Computations bottleneck. We now consider that communications are instantaneous. At a given time t, due to the form of the local functions, the only nodes that can increase $k_j(t)$ (and so improve the error for dimension j) when $k_j(t)$ is odd are the nodes in Q. Thus, $k_j(t)$ is bounded by two times the number of times f_{ij} has been sampled by nodes in Q, which leads to:

$$\sum_{j=1}^{m} k_j(t) \leq 2\sum_{l=1}^{\lceil t\rceil}\sum_{i\in Q}\sum_{j=1}^{m}\mathbb{1}\{\zeta_i(l) = j\} \leq 2|Q|\lceil t\rceil, \tag{3.4.6}$$

since each evaluation takes time 1. Parallelism is very limited in this case because only nodes in Q actually contribute to the progress. We can then use Jensen inequality with the convex function $x \mapsto q^{2x}$ to write that $\frac{1}{m}\sum_{j=1}^m q^{2k_j(t)} \geq q^{\frac{2}{m}\sum_{j=1}^m k_j(t)}$. We average the bounds obtained with Equations (3.4.5) and (3.4.6) and obtain:

$$\frac{2}{nm}\mathbb{E}[\sum_{i=1}^{n}\sum_{j=1}^{m}\frac{\|\theta^t_{ij}-\theta^*_{ij}\|^2}{\|\theta^0_{ij}-\theta^*_{ij}\|^2}] \geq q^{\frac{4|Q|\lceil t\rceil}{m}} + q^{2+\frac{2\lceil t\rceil}{\Delta\tau}}. \tag{3.4.7}$$

Then, we use that that $\kappa_s = m\kappa_b$ in this case, and we pick Δ as the diameter of the graph (or any constant fraction of the diameter, which only changes constant factors) and $Q = \{u\}$ where $u \in \arg\max_{u'}\max_v d(u', v)$ where $d(u, v)$ is the distance between nodes u and v in the graph $\mathcal{G}$. □

This bound can be further simplified into the asymptotic expression below:

Corollary 4 (Centralized lower bound). *Under the assumptions of Theorem 12, there exist functions such that for any black-box procedure, the time to reach a precision ε is lower*

bounded by:

$$\Omega\left(\left[m+\sqrt{m\kappa_s}+\tau\Delta\sqrt{\kappa_b}\right]\log(\varepsilon^{-1})\right).$$

The previous lower bounds rely on the diameter of the network, without assuming any structure. We use in this section the same trick as in Scaman et al. [2017b] to extend the lower bounds to the gossip communications setting.

COROLLARY 5 (Decentralized lower bound). *Let $\gamma > 0$, and $\kappa_b > 0$. There exist a gossip matrix W with spectral gap γ and $n \times m$ functions $f_{ij} : \ell_2 \to \mathbb{R}$ such that each f_{ij} is convex and L_{ij}-smooth, $f_i \hat{=} \sum_{j=1}^m f_{ij}$ is L_i-smooth and σ_i-strongly convex with L_{ij} and σ_i such that $\kappa_b \geq L_i/\sigma_i \geq L_{ij}/\sigma_i$ for all i, j, and such that if f_i is the local function of node i then for any black-box procedure, the time to reach precision ε is lower bounded by:*

$$\Omega\left(\left[m+\sqrt{m\kappa_s}+\tau\sqrt{\frac{\kappa_b}{\gamma}}\right]\log(\varepsilon^{-1})\right).$$

PROOF. The proof relies on the fact that for all $\gamma > 0$, it is possible to construct a gossip matrix on a line graph of size n with spectral gap γ. In this case, the diameter of the graph is n, which is of order $\gamma^{-1/2}$. Details can be found in Theorem 2 Scaman et al. [2017b]. □

It is interesting to remark that considering the finite-sum setting only changes the lower bound on the computation cost. This is not surprising since it only allows to compute cheaper stochastic gradients but cannot reduce communication cost without additional assumptions on the functions used. As a matter of fact, the computation and communication aspects are treated separately in the lower bound. This could suggest room for improvement for this lower bound. Yet, the bound we obtain is actually tight in the sense that it is matched by the ADFS-Synch algorithm. Similarly, the black-box model is not constrained to communicate using W. Nevertheless, the result is tight in the sense that the bound is achieved by an algorithm (ADFS-Synch) that communicates using only W, and so a better lower bound cannot be obtained by considering a weaker communication model. However, considering other communication models could potentially allow to state a stronger version of Corollary 5 by inverting the quantifiers, and having that for *any* gossip matrix W with spectral gap γ, there exist functions f_{ij} such that the rest holds.

There is actually a small subtle gap between the lower and the upper bounds, which is caused by the fact that the communication lower bound depends on κ_b, whereas the complexity of ADFS-Synch depends on κ_{comm}, which can be bigger. Yet, we show in Corollary 8 that $\kappa_{\text{comm}} = O(\kappa_b)$ in the case of the worst case function used for the lower bound. More generally $\kappa_{\text{comm}} = O(\kappa_b)$ for generalized linear models such as linear regression when the regularization parameter is the same for all nodes, which is a prime use-case for ADFS.

3.4.3. Linear speedup. Linear speedup, *i.e.*, dividing the running time of an algorithm by $O(n)$ when using n nodes, is a key property in distributed optimization. In the non-smooth setting, the (tight) lower bound [Scaman et al., 2018] is of the form $\frac{R\tau}{\varepsilon\sqrt{\gamma}} + \frac{R^2}{\varepsilon^2}$ and so for small enough ε, convergence does not depend on τ or γ, and so linear speedup is achieved since subgradients are computed n times faster with n nodes. In both the convex [Xu et al., 2020a] and strongly convex [Scaman et al., 2017b] settings, lower bounds are of the form $(1 + \tau/\sqrt{\gamma})R(\varepsilon)$, where $R(\varepsilon)$ is a rate that is independent of the network size or topology. Thus, linear speedup can also be obtained provided communications are fast

enough ($\tau/\sqrt{\gamma} < 1$). We show a similar phenomenon in the distributed finite sum setting, in which linear speedup is achieved as long as $\tau\sqrt{\kappa_b/\gamma} < m + \sqrt{m\kappa_s}$. Note however that for smooth problems, it is best to talk about *network independence* [Pu et al., 2020] rather than linear speedup, since the running time does not increase with the number of nodes, but the datasets processed are n times bigger. Yet, optimal single-machine variance-reduced algorithms on the whole dataset enjoy a $nm + \sqrt{nm\kappa_s}\log(1/\varepsilon)$ complexity [Lan and Zhou, 2017], so processing datasets that are n times bigger is not necessarily n times longer, and thus *distributed algorithms cannot achieve linear speedup compared to the best single machine algorithm on the same dataset.*

Note that this does not contradict the linear speedup obtained by Katyusha [Allen-Zhu, 2017], which considers a fixed number of samples processed by an increasing number of nodes, which corresponds to the case in which all nodes have access to the whole (replicated) dataset. In particular, the bound of Theorem 12 can be weakened to match the Katyusha complexity results when all nodes are forced to have the same local functions. Indeed, very few nodes actually contribute to reducing the error in the worst case (Theorem 12), whereas this cannot happen if all nodes have the same local function. Similarly, these results change if one considers stochastic gradient evaluations, without the finite sum structure. In this case, sampling n times more stochastic gradient eventually divides the variance by n (see Davies et al. [2020] for a discussion of how to do so with minimal communication), and so ultimately yields linear speedup [Pu et al., 2020, Koloskova et al., 2020]. As in the non-smooth case, network-related constants only affect higher-order terms, and so linear speedup is guaranteed provided ε is small enough.

3.5. Block Accelerated Proximal Coordinate Gradient with Arbitrary Sampling

Before we start with the actual distributed algorithm, we first introduce a coordinate descent method, which we will then apply to a well-chosen dual formulation to derive ADFS. The convergence results of ADFS are based on the convergence of this Accelerated Proximal Coordinate Gradient method. ADFS is derived in a way that is similar to that of the classical APCG algorithm [Lin et al., 2015b], but we integrate the decentralized aspect, which requires several improvements over the original APCG.

3.5.1. General formulation. In this section, we study the generic problem of accelerated proximal coordinate descent. We give an algorithm that works with *arbitrary sampling* of *blocks* of coordinates of arbitrary size, thus yielding a stronger result than state-of-the-art approaches [Fercoq and Richtárik, 2015, Lin et al., 2015b]. This is a key contribution that allows to obtain fast rates when sampling probabilities are heterogeneous and determined by the problem. In the dual formulation of the problem, there is one coordinate per point in the dataset as well as one for each edge of the network. Therefore, the block aspect allows to have a synchronous algorithm by picking only coordinates of a given kind (data point or network edge) to perform computation and communication rounds. Similarly, arbitrary sampling is useful to pick different probabilities for computing and for communicating. To avoid any confusion with the rest of the paper, we note d the dimension of the problem that we wish to solve. More specifically, we study the following generic problem, where $x^{(i)}$ is the

i-th coordinate of vector $x \in \mathbb{R}^d$:

$$\min_{x \in \mathbb{R}^d} \quad q_A(x) + \sum_{i=1}^{d} \psi_i(x^{(i)}), \tag{3.5.1}$$

where all the functions ψ_i are convex and q_A is such that there exists a matrix A such that q_A is (σ_A)-strongly convex on $\mathrm{Ker}(A)^\perp$, the orthogonal of the kernel of A, as defined by Equation (3.5.2). For the problems that we consider, $\mathrm{Ker}(A)^\perp \subsetneq \mathbb{R}^d$ and so q_A is not strongly convex on the whole space. We introduce matrix A in order to recover the benefits of strong convexity, but only on a subspace. We note $A^\dagger$ is the pseudo-inverse of A, meaning that $A^\dagger A$ is the projector on $\mathrm{Ker}(A)^\perp$. We sometimes abuse notations by writing $A^{-\frac{1}{2}}$ instead of $(A^\dagger)^{\frac{1}{2}}$. The strong convexity on $\mathrm{Ker}(A)^\perp$ can be written as the fact that for all $x, y \in \mathbb{R}^d$:

$$q_A(x) - q_A(y) \geq \nabla q_A(y)^\top A^\dagger A(x-y) + \tfrac{\sigma_A}{2}(x-y)^\top A^\dagger A(x-y). \tag{3.5.2}$$

Note that this implies that q_A is constant on $\mathrm{Ker}(A)$, so in particular there exists a function q such that for any $x \in \mathbb{R}^d$, $q_A(x) = q(Ax)$. In this case, σ_A is such that $x^\top A^\top \nabla^2 q(y) Ax \geq \sigma_A \|x\|^2$ for any $x \in \mathrm{Ker}(A)^\perp$ and $y \in \mathbb{R}^d$. Besides, q_A is assumed to be (M)-smooth on $\mathrm{Ker}(A)^\perp$, meaning that there exists a matrix M such that:

$$q_A(x) - q_A(y) \leq \nabla q_A(y)^\top A^\dagger A(x-y) + \tfrac{1}{2}(x-y)^\top M(x-y). \tag{3.5.3}$$

The block-version of APCG with arbitrary sampling is presented in Algorithm 3, and we explicit its rate in Theorem 13.

3.5.2. Algorithm and results. In this section, we denote $e_i \in \mathbb{R}^d$ the unit vector corresponding to coordinate i, and so $x^{(i)} = e_i^\top x$ for any $x \in \mathbb{R}^d$. Let $R_i = e_i^\top A^\dagger A e_i$ and p_i be the probability that coordinate i is picked to be updated. For a batch of coordinates $b \subset \{1, \cdots, d\}$, we introduce the random matrix P_b which is a *diagonal* matrix such that $(P_b)_{ii} = p_i$ if $i \in b$ and $(P_b)_{ii} = 0$ otherwise, where $p_i = \sum_{b,\ i \in b} p_b$ if p_b is the probability of sampling block b. In particular, $\mathbb{E}\left[P_b^\dagger\right] = Id$, where $P_b^\dagger$ is the pseudo-inverse of P_b. The matrix P_b defines the sampling that is performed. This allows to have a flexible sampling with blocks of arbitrary sizes sampled with arbitrary probabilities. Constant S is such that $S^2 \geq \lambda_{\max}(A^\dagger A P_b^\dagger M P_b^\dagger A^\dagger A)$ for all batches b, where we recall that M is the smoothness matrix of function q_A, as defined in Equation (3.5.3). This definition of S is rather hard to read, but one can see it as the max over b of the smooothness of q_A in the direction given by P_b, divided by p_b^2. This then allows to set p_b so that non-smooth directions are sampled more often, and so $\lambda_{\max}(A^\dagger A P_b^\dagger M P_b^\dagger A^\dagger A)$ is the same for all b. Then, following the approach of Nesterov and Stich [Nesterov and Stich, 2017], we fix $A_0, B_0 \in \mathbb{R}^+$ and recursively define the sequences $\alpha_t, \beta_t, a_t, A_t$ and B_t such that:

$$a_{t+1}^2 S^2 = A_{t+1} B_{t+1}, \qquad B_{t+1} = B_t + \sigma_A a_{t+1}, \qquad A_{t+1} = A_t + a_{t+1},$$
$$\alpha_t = \frac{a_{t+1}}{A_{t+1}}, \qquad \beta_t = \frac{\sigma_A a_{t+1}}{B_{t+1}}.$$

Finally, we introduce the sequences (y_t), (v_t) and (x_t), that are all initialized at 0. We define $\eta_t = \frac{a_{t+1}}{B_{t+1}}$ and the proximal operator $\mathrm{prox}_{\eta f}$ is defined in Equation (3.2.1).

For generalized APCG to work well, the proximal operator needs to be taken in the subspace defined by the projector $A^\dagger A$, and so the non-smooth ψ_i terms have to be separable after composition with $A^\dagger A$. Since $A^\dagger A$ is a projector, this constraint is equivalent to stating

Algorithm 3 Generalized APCG(A_0, B_0, S, σ_A)

$y_0 = 0$, $v_0 = 0$, $t = 0$
while $t < T$ **do**
  $y_t = \frac{(1-\alpha_t)x_t + \alpha_t(1-\beta_t)v_t}{1-\alpha_t\beta_t}$
  Sample b_t with probability p_{b_t}
  $v_{t+1} = v_{t+\frac{1}{2}} = (1-\beta_t)v_t + \beta_t y_t - \eta_t P_b^\dagger \nabla q_A(y_t)$
  $v_{t+1}^{(i)} = \text{prox}_{\eta_t p_i^{-1}\psi_i}\left(v_{t+\frac{1}{2}}^{(i)}\right)$ for all $i \in b$
  $x_{t+1} = y_t + \alpha_t P_b^\dagger A^\dagger A(v_{t+1} - (1-\beta_t)v_t - \beta_t y_t)$

that either $R_i = 1$ (projection does not affect the coordinate i), or $\psi_i = 0$ (no proximal update to make).

ASSUMPTION 4. *The functions q_A and ψ are such that Equation (3.5.2) holds for some $\sigma_A \geq 0$ and Equation (3.5.3) holds for some M. Besides, ψ and A are such that either $R_i = 1$ or $\psi_i = 0$ for all $i \in \{1, ..., d\}$.*

This natural assumption allows us to formulate the proximal update in standard squared norm since the proximal operator is only used for coordinates i for which $A^\dagger A e_i = e_i$. Then, we formulate Algorithm 3 and analyze its rate in Theorem 13.

THEOREM 13. *Let $F : x \mapsto q_A(x) + \sum_{i=1}^d \psi_i\left(x^{(i)}\right)$ such that Assumption 4 holds. If S is such that $S^2 \geq \lambda_{\max}((A^\dagger A P_b^\dagger M P_b^\dagger A^\dagger A)$ for all b and $1 - \beta_t - \frac{\alpha_t}{p_i} \geq 0$ for all i such that $\psi_i \neq 0$, the sequences v_t and x_t generated by APCG verify:*

$$B_t \mathbb{E}[\|v_t - \theta^\star\|_{A^\dagger A}^2] + 2A_t\left[\mathbb{E}[F(x_t)] - F(\theta^\star)\right] \leq C_0,$$

where $C_0 = B_0\|v_0 - \theta^\star\|^2 + 2A_0\left[F(x_0) - F(\theta^\star)\right]$ and $\theta^\star$ is a minimizer of F. The rate of APCG depends on S through the sequences α_t and β_t.

PROOF. The proof is an adaptation of the proofs from Lin et al. [2015b] and Nesterov and Stich [2017]. In particular, the structure is similar to that of Nesterov and Stich [2017]. The difference is that the $\|v_{t+1} - \theta^\star\|^2$ is studied in norm $A^\dagger A$ and that v_{t+1} cannot be expressed simply as v_t minus a gradient term the way it was before because of the proximal update. Therefore, we develop $\|v_{t+1} - \theta^\star\|_{A^\dagger A}^2$ using the strong convexity of the proximal mapping instead, which is a key argument from Lin et al. [2015b], thus leading to the following base inequality, which is a block version of Hendrikx et al. [2019b, Equation (11)] that can be derived using the same arguments:

$$\begin{aligned}
&\frac{1}{2\eta_t}[\|v_{t+1} - \theta^\star\|_{A^\dagger A}^2 + \|v_{t+1} - w_t\|_{A^\dagger A}^2 - \|\theta^\star - w_t\|_{A^\dagger A}^2] \\
&\qquad \leq \langle P_b^\dagger \nabla q_A(y_t), \theta^\star - v_{t+1}\rangle_{A^\dagger A} + \sum_{i \in b} \frac{1}{p_i}\left[\psi_i\left(\theta^{\star(i)}\right) - \psi_i\left(v_{t+1}^{(i)}\right)\right].
\end{aligned}$$

The other key point of the APCG proof is that x_t can be expressed as a convex combination of all the v_l for $l \leq t$. This does not directly extend to the arbitrary sampling case because the coefficients may not be the same for all coordinates, so we need to prove that the convex combination property holds separately for each coordinate. This is possible because the only

terms that depend on the coordinates in the decomposition of x_t come from the $v_{t+1} - w_t$ term, where $w_t = (1-\beta_t)v_t + \beta_t y_t$. Yet, $v_{t+1} = w_t$ when the coordinate is not picked, so we can still write that $x_{t+1}^{(i)} = y_t^{(i)} + \frac{\alpha_t}{p_i}(v_{t+1}^{(i)} - w_t^{(i)})$ even when coordinate i is not picked at time t. Hendrikx et al. [2019b, Lemma 1]. The complete proof is given in Appendix 3.5.1.

□

3.5.3. Explicit rates. Algorithm 3 is a general method that requires to set values for A_0, B_0, α_0 and β_0. The two following corollaries give choices of parameters depending on whether $\sigma_A > 0$ or $\sigma_A = 0$, along with the rate of APCG in these cases.

COROLLARY 6 (Strongly Convex case). *Let F verify the assumptions of Theorem 13. If $\sigma_A > 0$, we choose for all $t \in \mathbb{N}$ $\alpha_t = \beta_t = \rho$ and $A_t = \sigma_A^{-1} B_t = (1-\rho)^{-t}$ with $\rho = \sqrt{\sigma_A} S^{-1}$. In this case, the condition $1 - \beta_t - \frac{\alpha_t}{p_i} \geq 0$ can be weakened to $1 - \frac{\alpha_t}{p_i} \geq 0$ and it is automatically satisfied by our choice of S, α_t and β_t. Denoting $C_0 = \sigma_A \|v_0 - \theta^\star\|^2 + 2\left[F(x_0) - F(\theta^\star)\right]$, the sequences x_t and v_t verify:*

$$\sigma_A \mathbb{E}[\|v_t - \theta^\star\|^2_{A^\dagger A}] + 2\left[\mathbb{E}[F(x_t)] - F(\theta^\star)\right] \leq C_0 (1-\rho)^t.$$

Corollary 6 is the extension of the results of Lin et al. [2015b] to block coordinates and arbitrary sampling. In particular, APCG converges linearly in this case, and we recover the rate of Lin et al. [2015b] in the special case in which we choose blocks of size 1 uniformly at random. Note that an arbitrary sampling extension of accelerated coordinate descent was already present in Hanzely and Richtárik [2019] but without the block or proximal aspects on which our technical contributions are focused.

COROLLARY 7 (Convex case). *Let F verify the assumptions of Theorem 13. If $\sigma_A = 0$, we choose $\beta_t = 0$ and $\alpha_0 = p_{\min}^2$ with $p_{\min} = \min_{i:\psi_i \neq 0} p_i$. In this case, the condition $1 - \beta_t - \frac{\alpha_t}{p_i} \geq 0$ is always satisfied for our choice of S and the error verifies:*

$$\mathbb{E}[F(x_t)] - F(\theta^\star) \leq \frac{2}{t^2}\left[S^2 r_t^2 + \frac{2}{p_{\min}^2}\left[F(x_0) - F(\theta^\star)\right]\right],$$

with $r_t^2 = \|v_0 - \theta^\star\|^2_{A^\dagger A} - \mathbb{E}[\|v_t - \theta^\star\|^2_{A^\dagger A}]$. Note that there is no need to choose parameters A_t and B_t since only parameter α_t is required in this case.

PROOF OF COROLLARY 7. Let $B_t = B_0$ for some $B_0 > 0$. This allows to write $(A_{t+1} - A_t)^2 S^2 = A_t B_0$ for all t, which is a second degree polynomial in the variable A_{t+1}. We choose the positive root in order to have $a_{t+1} \geq 0$, which yields $A_{t+1} = A_t + \frac{B_0}{2S^2}(1 + (1 + 4S^2 B_0^{-1} A_t)^{\frac{1}{2}})$. From there, we can deduce the expression of a_{t+1} and that of α_t in terms of A_t. This leads after some simplifications to: $\alpha_{t+1}^{-2} - \alpha_{t+1}^{-1} - \alpha_t^{-2} = 0$, and so $\alpha_{t+1}(\sqrt{\alpha_t^4 + 4\alpha_t^2} - \alpha_t^2)/2$, which is the exact same recursion as in Lin et al. [2015b] and Fercoq and Richtárik [2015]. In particular, only the value of α_0 matters and only the sequence α_t actually needs to be computed, since the only coefficients needed are the α_t and $\frac{a_{t+1}}{B_{t+1}} = \frac{1}{\alpha_t S^2}$.

We would like to choose the highest possible α_0, such that $1 - \alpha_t / p_{\min} \geq 0$, so we take $\alpha_0 = p_{\min}$, which is enough since (α_t) is a decreasing sequence. This leads to $A_0 = [(2/p_{\min} - 1)^2 - 1] B_0 / 4S^2 \leq B_0 / (p_{\min} S)^2$. Since $A_0 \geq 0$, a direct recursion yields $A_t \geq B_0 t^2 / (4S^2)$.

With $\Delta F_t = \mathbb{E}[q_A(x_t) + \psi(x_t)] - q_A(\theta_A^\star) + \psi_A(\theta_A^\star)$, then:

$$\Delta F_t \leq \frac{1}{2A_t}\left(B_0 r_t^2 + 2A_0 F_0\right) = \frac{B_0}{2A_t}\left(r_t^2 + \frac{2}{S^2 p_{\min}^2}\Delta F_0\right) \leq \frac{2S^2}{t^2}\left(r_t^2 + \frac{2}{S^2 p_{\min}^2}\Delta F_0\right).$$

which finishes the proof of the rate. □

Our extended APCG algorithm is closely related with an arbitrary sampling version of APPROX [Fercoq and Richtárik, 2015]. Similarly to Lee and Sidford [2013], APPROX also uses iterations that can be more efficient, especially in the linear case. These extensions can also be applied to APCG under the same assumptions, as shown by Lin et al. [2015b].

3.6. Accelerated Decentralized Stochastic Algorithm

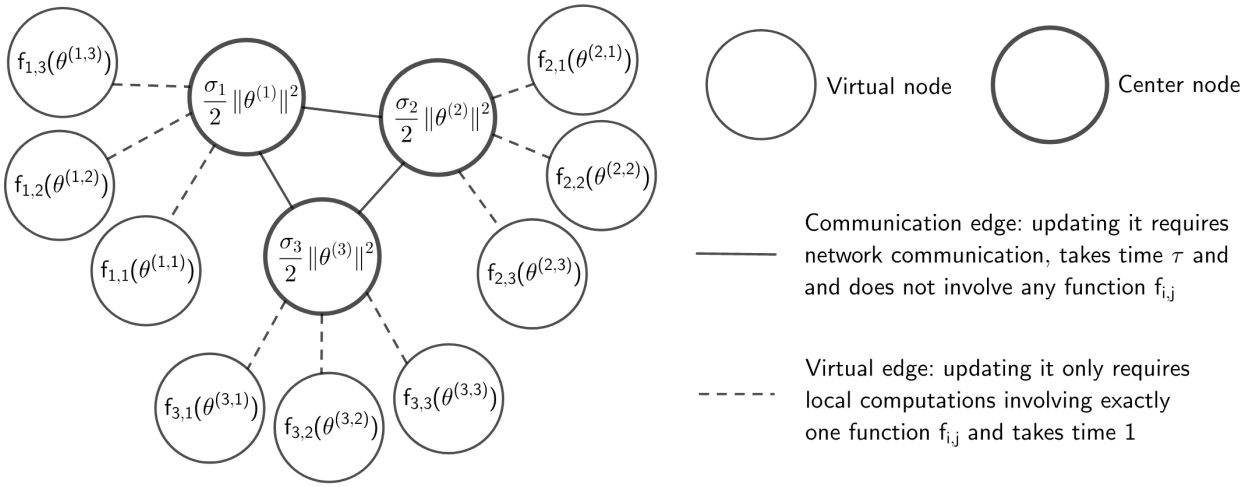


FIGURE 1. Illustration of the augmented graph for $n = 3$ and $m = 3$.

3.6.1. The dual problem. We now specify our approach to solve the problem of Equation (3.1.1). The first (classical) step consists in considering that all nodes have a local parameter, but that all local parameters should be equal because the goal is to have the global minimizer of the sum. Therefore, the problem writes:

$$\min_{\theta \in \mathbb{R}^{n\times d}} \quad \sum_{i=1}^{n} f_i(\theta^{(i)}) \quad \text{such that} \quad \theta^{(i)} = \theta^{(j)} \text{ if } j \in \mathcal{N}(i), \tag{3.6.1}$$

where $\mathcal{N}(i)$ represents the neighbors of node i in the communication graph. Then, ES-DACD and MSDA are obtained by applying accelerated (coordinate) gradient descent to an appropriate dual formulation of Problem (3.6.1). In the dual formulation, constraints become variables and so updating a dual coordinate consists in performing an update along an edge of the network. In this work, we consider a new virtual graph in order to get a stochastic algorithm for finite sums. The transformation is sketched in Figure 1, and consists in replacing each node of the initial network by a star network. The centers of the stars are connected by the actual communication network, and the center of the star network replacing node i has the local function $f_i^{\text{comm}} : x \mapsto \frac{\sigma_i}{2}\|x\|^2$. The center of node i is then connected with m nodes whose local functions are the functions f_{ij} for $j \in \{1, ..., m\}$. Denoting E the number of edges of the initial graph, then the augmented graph has $n(1 + m)$ nodes and

$E + nm$ edges. This augmented graph formulation was introduced in the conference version of this paper [Hendrikx et al., 2019b].

Then, we consider one parameter vector $\theta^{(ij)}$ for each function f_{ij} and one vector $\theta^{(i)}$ for each function f_i^{comm}. Therefore, there is one parameter vector for each node in the augmented graph. We impose the standard constraint that the parameter of each node must be equal to the parameters of its neighbors, but neighbors are now taken in the augmented graph. This yields the following minimization problem:

$$\min_{\theta \in \mathbb{R}^{n(1+m)d}} \sum_{i=1}^{n} \left[\sum_{j=1}^{m} f_{ij}(\theta^{(ij)}) + \frac{\sigma_i}{2} \|\theta^{(i)}\|^2 \right] \\ \text{such that } \theta^{(i)} = \theta^{(j)} \text{ if } j \in \mathcal{N}(i), \text{ and } \theta^{(ij)} = \theta^{(i)} \ \ \forall j \in \{1, .., m\}. \tag{3.6.2}$$

In the rest of the paper, we use letters k, ℓ to refer to any nodes in the augmented graph, and letters i, j to specifically refer to a communication node and one of its virtual nodes. More precisely, we denote $(k\ell)$ the edge between the nodes k and ℓ in the augmented graph. Note that k and ℓ can be virtual or communication nodes. To clearly make the distinction between node variables and edge variables, for any vector on the set of nodes of the augmented graph $x \in \mathbb{R}^{n(1+m)d}$ and for $k \in \{1, ..., n(1+m)\}$, we write $x^{(k)} \in \mathbb{R}^d$ (superscript notation) the subvector associated with node k. Similarly, for any vector on the set of edges of the augmented graph $\lambda \in \mathbb{R}^{(E+nm)d}$ and for any edge $(k\ell)$ we write $\lambda_{k\ell} \in \mathbb{R}^d$ (subscript notation) the vector associated with edge $(k\ell)$. For node variables, we use the subscript notation with a t to denote time, for instance in Algorithm 4. By a slight abuse of notations, we use indices (ij) instead of $(k\ell)$ when specifically referring to virtual edges (or virtual nodes) and denote λ_{ij} instead of $\lambda_{i,(ij)}$ the virtual edge between node i and node (ij) in the augmented graph. We note $e^{(k)} \in \mathbb{R}^{n(1+m)}$ the unit vector associated with node k and $e_{k\ell} \in \mathbb{R}^{E+nm}$ the unit vector associated with edge $k\ell$. We denote $M_1 \otimes M_2$ the Kronecker product between matrices M_1 and M_2.

Constraints matrix. The constraints of Problem (3.6.2) can be rewritten $A^\top \theta = 0$ in matrix form $A \in \mathbb{R}^{n(1+m)d \times (nm+E)d}$ is such that for any $x \in \mathbb{R}^d$,

$$A(e_{k\ell} \otimes x) = \mu_{k\ell}[(e^{(k)} - e^{(\ell)}) \otimes P_{k\ell} x],$$

for some $\mu_{k\ell} > 0$, and where $P_{k\ell}$ is a projector. For communication edges, we choose $P_{k\ell} = I_d$, and for virtual edges, we choose P_{ij} such that f_{ij} is L_{ij}-smooth with respect to P_{ij}, in the sense that $\nabla^2 f_{ij}(x) \preccurlyeq L_{ij} P_{ij}$ for all x (but P_{ij} does not need to be full rank). Note that this implies that f_{ij}^* is $(1/L_{ij})$-strongly convex on $\text{Ker}(P_{ij})^\perp$ and infinite elsewhere. The matrix A is therefore completely defined by the $\mu_{k\ell}$ and the P_{ij}. Most results in the following sections heavily depend on the matrix A and it is therefore very important to understand its structure. In particular, decentralized communications are defined by the matrix A. Indeed, A can be understood as the canonical square root of the weighted Laplacian of the augmented graph. Similarly, if we note $A_{\text{comm}} \in \mathbb{R}^{n \times E}$ the restriction of A to non-virtual edges then A_{comm} is a square root of the weighted Laplacian of the communication graph. This is why a rescaled version of $A_{\text{comm}} A_{\text{comm}}^\top \in \mathbb{R}^{n \times n}$ is used as the gossip matrix in Algorithm 4. To make things clearer, A can be written as:

$$A = \begin{pmatrix} A_{\text{comm}} \otimes I_d & D_\mu \\ 0 & -D_\mu^{\text{diag}} \end{pmatrix}, \text{ with } D_\mu^{\text{diag}} = \begin{pmatrix} \mu_{11} P_{11} & 0 & 0 \\ 0 & \cdots & 0 \\ 0 & 0 & \mu_{nm} P_{nm} \end{pmatrix} \text{ and}$$

$$D_\mu = \begin{pmatrix} \mu_{11}P_{11} & \cdots & \mu_{1m}P_{1m} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \cdots & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \mu_{n1}P_{n1} & \cdots & \mu_{nm}P_{nm} \end{pmatrix} \in \mathbb{R}^{nd \times nmd}.$$

All *communication nodes* are linked by the true graph, whereas all *virtual nodes* are linked to their corresponding communication node. Note that A is defined differently in the conference paper [Hendrikx et al., 2019b]. Although the new definition of A as an $n(m+1)d \times (E+nm)d$ matrix is heavier in terms of notations, it allows to derive a much better communication complexity in some cases, for instance when f_{ij} is a generalized linear model. Now that we have defined the matrix A and emphasized its importance, we can write the dual formulation of the problem as:

$$\max_{\lambda \in \mathbb{R}^{(nm+E)d}} -\sum_{i=1}^{n} \left[\sum_{j=1}^{m} f_{ij}^* \left((A\lambda)^{(ij)} \right) + \frac{1}{2\sigma_i} \|(A\lambda)^{(i)}\|^2 \right], \tag{3.6.3}$$

where the parameter λ is the Lagrange multiplier associated with the constraints of Problem (3.6.2)—more precisely, for an edge $(k\ell)$, $\lambda_{k\ell} \in \mathbb{R}^d$ is the Lagrange multiplier associated with the constraint $\mu_{k\ell}P_{k\ell}(\theta^{(k)} - \theta^{(\ell)}) = 0$. This critically relies on the fact that $f_{ij}^*(P_{ij}x) = f_{ij}^*(x)$ for all $x \in \mathrm{dom}(f_{ij}^*) = \mathrm{Ker}(P_{ij})^\perp$. At this point, the functions f_{ij} are only assumed to be convex (and not necessarily strongly convex) meaning that the functions f_{ij}^* are potentially non-smooth. This problem could be bypassed by transferring some of the quadratic penalty from the communication nodes to the virtual nodes before going to the dual formulation. Yet, this approach fails when m is large because the smoothness parameter of f_{ij}^* would scale as m/σ_i at best, whereas a smoothness of order $1/\sigma_i$ is required to match optimal finite-sum methods. A better option is to consider the f_{ij}^* terms as non-smooth and perform proximal updates on them. The rate of proximal gradient methods such as APCG [Lin et al., 2015b] does not depend on the strong convexity parameter of the non-smooth functions f_{ij}^*. Recall that each f_{ij}^* is $(1/L_{ij})$-strongly convex with respect to P_{ij}, so we can rewrite the previous equation in order to transfer all the strong convexity to the communication node. Noting that $(A\lambda)^{(ij)} = -\mu_{ij}\lambda_{ij}$ when node (ij) is a virtual node associated with node i, we rewrite the dual problem as:

$$\min_{\lambda \in \mathbb{R}^{(E+nm)d}} q_A(\lambda) + \sum_{i=1}^{n} \sum_{j=1}^{m} \psi_{ij}(\lambda_{ij}), \tag{3.6.4}$$

with $\psi_{ij} : x \mapsto \widetilde{f_{ij}^*}(-\mu_{ij}x)$ and $\widetilde{f_{ij}^*} : x \mapsto f_{ij}^*(x) - \frac{1}{2L_{ij}}\|x\|_{P_{ij}}^2$ and $q_A : x \mapsto \frac{1}{2}x^\top A^\top \Sigma^\dagger A x$, where Σ is the diagonal matrix such that the upper left (communication) block is equal to $\mathrm{diag}(\sigma_1, \cdots, \sigma_n) \otimes I_d$, and the rest of the diagonal is made of the blocks $L_{ij}P_{ij}$ for the virtual node (ij). Since dual variables are associated with edges, using coordinate descent algorithms on dual formulations from a well-chosen augmented graph of constraints allows us to handle both computations and communications in the same framework. Indeed, choosing a variable corresponding to an actual edge of the network results in a communication along this edge, whereas choosing a virtual edge results in a local computation step. Then, we balance the ratio between communications and computations by simply adjusting the probability of picking a given kind of edges.

3.6.2. The Algorithm: ADFS Iterations and Expected Error. Recall that we would like to solve the problem of Equation (3.6.4), which is to optimize the sum of a

smooth and strongly convex term and of a non-smooth convex separable term. Proximal coordinate gradient algorithms are known to work well for these problems, which is why we would like to use APCG [Lin et al., 2014]. Yet, the following points would lead to suboptimal rates if the standard APCG algorithm were used directly:

(1) The function q_A is strongly-convex only on $\mathrm{Ker}(A)^\perp$.
(2) Picking blocks of coordinates is required to obtain a synchronous algorithm.
(3) Choosing different probabilities for computation and communication coordinates allows to balance the ratio between communication and computation.

These 3 points show the need for extending APCG and motivate our assumptions for Algorithm 3. Applying it to the problem of Equation (3.6.4) yields the general ADFS algorithm. We denote $W_{k\ell} \in \mathbb{R}^{n(1+m)\times n(1+m)}$ the matrix such that $W_{k\ell} = (e^{(k)} - e^{(\ell)})(e^{(k)} - e^{(\ell)})^\top$ for any edge $(k\ell)$. In the previous section, we considered the problem variables to be vectors in $\mathbb{R}^{n(1+m)d}$ in order to define the right matrix A. Yet, variables x_t, y_t and v_t from Algorithm 4 are variables associated with the nodes of the augmented graph and we will therefore consider them as matrices in $\mathbb{R}^{n(1+m)\times d}$ (one row for each node) instead of vectors, which greatly simplifies notations. These variables are obtained by multiplying the dual variables of the proximal coordinate gradient algorithm applied to the dual problem of Equation (3.6.4) by A on the left. We denote $\sigma_A = \lambda^+_{\min}(A^\top \Sigma^\dagger A)$ the smallest non-zero eigenvalue of the matrix $A^\top \Sigma^\dagger A$. Algorithm 4 depends on an undefined parameter $\rho > 0$ at this point. Yet, Theorem 14 gives a condition on ρ that ensures convergence, and we give optimal choices of ρ for specific settings later (Lemma 11).

Algorithm 4 ADFS$(A, (\sigma_i), (L_{ij}), (\mu_{k\ell}), (p_{k\ell}), \rho)$

$\sigma_A = \lambda^+_{\min}(A^\top \Sigma^\dagger A)$, $\eta = \frac{\rho}{\sigma_A}$, $W_b = AP_b^\dagger A^\top$, $\widetilde{W}_b = AP_b^\dagger A^\dagger$.
$x_0 = y_0 = v_0 = z_0 = 0^{(n+nm)\times d}$ *// Initialization*
for $t = 0$ to $K-1$ **do** *// Run for K iterations*
  $y_t = \frac{1}{1+\rho}(x_t + \rho v_t)$
  Sample block of edges b *// Edges sampled from the augmented graph*
  $z_{t+1} = v_{t+1} = (1-\rho)v_t + \rho y_t - \eta W_b \Sigma^\dagger y_t$ *// (Virtual) Communication using W_b*
  if b is a block of virtual edges **then**
    for $i = 1$ to n **do**
      for j such that $(ij) \in b$ **do**
        $v_{t+1}^{(ij)} = \mathrm{prox}_{\eta\mu_{ij}^2 p_{ij}^{-1} \widetilde{f}_{ij}^*}\left(z_{t+1}^{(ij)}\right)$ *// Virtual node update using f_{ij}*
      $v_{t+1}^{(i)} = z_{t+1}^{(i)} + \sum_{j,(ij)\in b}(z_{t+1}^{(ij)} - v_{t+1}^{(ij)})$ *// Center node update*
  $x_{t+1} = y_t + \rho\widetilde{W}_b(v_{t+1} - (1-\rho)v_t - \rho y_t)$
return $\theta_K = \Sigma^\dagger v_K$ *// Return primal parameter*

THEOREM 14. *We denote $\theta^\star$ the minimizer of $F : x \mapsto \sum_{i=1}^n f_i(x)$ and $\theta_A^\star$ a minimizer of $F_A^* = q_A + \psi$. Then θ_t as output by Algorithm 4 verifies:*

$$\mathbb{E}[\|\theta_t - \theta^\star\|^2] \leq C_0(1-\rho)^t, \quad \text{if} \quad \rho^2 \leq \min_b \frac{\lambda^+_{\min}(A^\top\Sigma^\dagger A)}{\lambda_{\max}(A^\dagger A P_b^\dagger A^\top \Sigma^\dagger A P_b^\dagger A^\dagger A)}, \tag{3.6.5}$$

with $C_0 = \lambda_{\max}(A^\top \Sigma^{-2} A)\left[\|A^\dagger A\theta_A^\star\|^2 + 2\sigma_A^{-1}\left(F_A^*(0) - F_A^*(\theta_A^\star)\right)\right]$.

We now quickly discuss the convergence rate of ADFS, and present the basic derivations required to obtain Algorithm 4, as well as the proof Theorem 14.

Convergence rate. The parameter ρ controls the convergence rate of ADFS. It is defined by the minimum of the individual rates for each block, which involves the spectrum of a product of matrices related to the regularity of the local functions ($\Sigma^\dagger$), to the graph (A) and to the sampling scheme ($P_b^\dagger$). Note that Theorem 14 recovers the asynchronous version of ADFS Hendrikx et al. [2019b] if only one coordinate is sampled at each step. Relations are more complex in the general case, which is why simple scalar expressions are replaced by the spectrum of products of matrices in this paper. In Section 3.6.5, we carefully choose the free parameters $\mu_{k\ell}$ and $p_{k\ell}$ to get the best convergence speed.

Projection of virtual edges. We now verify that Assumption 4 is respected, so we can use Theorem 13 to derive Theorem 14. For any edge $(k\ell)$, either the proximal part $\psi_{k\ell} = 0$ or the dual coordinate is such that for all $\theta \in \mathbb{R}^d$, $(e_{k\ell}^\top \otimes \theta) A^\dagger A (e_{k\ell} \otimes \theta) = 1$, which is equivalent to having $A^\dagger A(e_{k\ell} \otimes \theta) = (e_{k\ell} \otimes \theta)$. In our case, $\psi_{k\ell} = 0$ when $(k\ell)$ is a communication edge. The condition is actually not verified for virtual edges in our formulation since we introduce the projectors P_{ij}. Yet, we do not need this to hold for any $\theta \in \mathbb{R}^d$. Indeed, the updates of Algorithm 4 are such that $v_t^{(ij)} \in \mathrm{Ker}(P_{ij})^\perp$ for all t and (ij), so we only need $A^\dagger A(e_{ij} \otimes \theta) = (e_{ij} \otimes \theta)$ to hold for $\theta \in \mathrm{Ker}(P_{ij})^\perp$. Lemma 8 shows that the projection condition is satisfied by virtual edges.

LEMMA 8. *$A^\dagger A(e_{ij} \otimes \theta) = e_{ij} \otimes \theta$ for all virtual edges (ij) and $\theta \in \mathrm{Ker}(P_{ij})^\perp$.*

PROOF. Let $\theta \in \mathrm{Ker}(P_{ij})^\perp$, and $x \in \mathbb{R}^{E+nm}$ such that $A(x \otimes \theta) = 0$. From the definition of A, either $x = 0$ or the support of x is a cycle of the graph. Indeed, for any edge $(k\ell)$, $A(e_{k\ell} \otimes \theta)$ has non-zero weights only on nodes k and ℓ. Virtual nodes have degree one, so virtual edges are part of no cycles and therefore $x^\top e_{k,\ell} = 0$ for all virtual edges $(k\ell)$. Operator $A^\dagger A$ is the projection operator on the orthogonal the kernel of A, so it is equal to P_{ij} on virtual edges, and $P_{ij}\theta = \theta$. □

Obtaining Line 6. The update of line 6 in Algorithm 4 comes from the fact that the update of block b writes $AP_b^\dagger \nabla q_A(y_t) = AP_b^\dagger A^\top \Sigma^\dagger y_t = W_b \Sigma^\dagger y_t$.

From edge variables to node variables. Algorithm 4 is obtained by directly applying Algorithm 3 on the dual problem of Equation (3.6.4). Then, all lines are multiplied by A on the left in order to switch from dual variables in $\mathbb{R}^{E+nm}$ associated with edges to primal variables in $\mathbb{R}^{n+nm}$ associated with nodes, which is a standard transformation [Scaman et al., 2017b, Hendrikx et al., 2019a]. Yet, APCG uses a proximal step, which is a non-linear operation, but the derivations can be adapted, as shown below. Thus, Algorithm 4 corresponds to applying Algorithm 3 to the Problem of Equation (3.6.4), which verifies Assumption 4, and so Theorem 14 is a corollary of Corollary 6.

We call $\tilde{v}_t$, $\tilde{y}_t$ and $\tilde{z}_t$ the dual variable sequences in $\mathbb{R}^{E+nm}$ obtained by applying Algorithm 3 on the dual problem of Equation 3.6.4. The new update equations can be retrieved by multiplying each line of Algorithm 3 by A on the left, so that for example $v_t = A\tilde{v}_t$. Yet, there is still a $\tilde{z}_{t+1}$ term because of the presence of the proximal update. More specifically, we write for the virtual edge between node i and its j-th virtual node:

$$\tilde{v}_{t+1}^T e_{ij} = \mathrm{prox}_{\eta_{ij}\psi_{ij}}\left(\tilde{z}_{t+1}^T e_{ij}\right). \tag{3.6.6}$$

Fortunately, this update only modifies $\tilde{v}_{t+1}$ when $\psi_{ij} \neq 0$. This means that z_{t+1} is only modified for local computation edges. Since local computation nodes only have one neighbour, the form of A ensures that for any $\tilde{z} \in \mathbb{R}^{n(1+m)}$ and virtual edge (k, ℓ) corresponding to node i and its j-th virtual node, $(A\tilde{z})^{(ij)} = -\mu_{k\ell}\tilde{z}_{k\ell}$. In particular, if node k is the center node i and node ℓ is the virtual node (ij), the proximal update can be rewritten:

$$\begin{aligned}
(A\tilde{v}_{t+1})^{(ij)} &= -\mu_{ij}\text{prox}_{\eta_{ij}\psi_{ij}}\left(-\frac{1}{\mu_{ij}}(A\tilde{z}_{t+1})^{(ij)}\right) \\
&= -\mu_{ij}\arg\min_v \frac{1}{2\eta_{ij}}\|v - \left(-\frac{1}{\mu_{ij}}(A\tilde{z}_{t+1})^{(ij)}\right)\|^2 + \psi_{ij}(v) \\
&= -\mu_{ij}\arg\min_v \frac{1}{2\eta_{ij}\mu_{ij}^2}\|-\mu_{ij}v - (A\tilde{z}_{t+1})^{(ij)}\|^2 + f_{ij}^*(-\mu_{ij}v) - \frac{\mu_{ij}^2}{2L_{ij}}\|v\|^2 \\
&= \arg\min_{\tilde{v}} \frac{1}{2\eta_{ij}\mu_{ij}^2}\|\tilde{v} - (A\tilde{z}_{t+1})^{(ij)}\|^2 + f_{ij}^*(\tilde{v}) - \frac{1}{2L_{ij}}\|\tilde{v}\|^2 \\
&= \text{prox}_{\eta_{ij}\mu_{ij}^2\tilde{f}_{ij}^*}\left((A\tilde{z}_{t+1})^{(ij)}\right),
\end{aligned}$$

where $\tilde{f}_{ij}^* : x \to f_{ij}^*(x) - \frac{1}{2L_{ij}}\|x\|^2$. For the center node, the update can be written:

$$\begin{aligned}
(A\tilde{v}_{t+1})^{(i)} &= (A\tilde{z}_{t+1})^{(i)} - \mu_{ij}e_{ij}^T\tilde{z}_{t+1} + \mu_{ij}\text{prox}_{\eta_{ij}\psi_{ij}}\left(-\frac{1}{\mu_{ij}}(A\tilde{z}_{t+1})^{(ij)}\right) \\
&= (A\tilde{z}_{t+1})^{(i)} + (A\tilde{z}_{t+1})^{(ij)} - \text{prox}_{\eta_{ij}\mu_{k\ell}^2\tilde{f}_{ij}^*}\left((A\tilde{z}_{t+1})^{(ij)}\right).
\end{aligned}$$

3.6.3. Implementation details. We now discuss implementation details for Algorithm 4, and provide its Python implementation in supplementary material.

Global information. The implementation of Algorithm 4 requires all nodes to have some global knowledge to be able to compute ρ, as defined in Equation (3.6.5). The two terms involved, $\lambda_{\min}^+(A^\top\Sigma^\dagger A)$ and $\max_b \lambda_{\max}(A^\dagger A P_b^\dagger A^\top\Sigma^\dagger A P_b^\dagger A^\dagger A)$, mix information about the topology of the graph and the regularity of local nodes. Yet, ρ can be computed beforehand, and only an upper bound is needed so one could use a rough approximation as long as all nodes agree on it before running the algorithm. Besides, ignoring the more subtle interactions with the graph topology, setting ρ only requires knowing the eigengap of the graph and the regularity of the local functions, which are typically assumed to be known quantities in decentralized optimization. Note that accelerated methods typically require knowledge of the smoothness and strong convexity constants of the problem (which includes the graph in our case) to run optimally [d'Aspremont et al., 2021]. Finally, although Algorithm 4 is presented from the point of view of a central coordinator, ADFS can be run in a fully decentralized way, as shown for synchronous communications in Algorithm 5. Nodes only need to agree on p_{comm} and on a seed at the beginning of the procedure to perform the communication steps of lines 8-13. Otherwise, each node draws one f_{ij} to perform a local stochastic proximal update. Without global synchronous rounds (as in the original conference paper), nodes need to know a global iterations counter in order to perform the convex combination steps. Yet, this can actually be relaxed using continuous-time contractions (but discrete gradient updates), as shown in Even et al. [2020].

Algorithm 5 Synch-ADFS, from the point of view of node i.

1: Choose parameters p_{ij} and μ_{ij} as in Assumption 5 (may use communication).
2: Compute and agree on parameters ρ, η and p_{comp} (requires communication).
3: $W = A_{\text{comm}} A_{\text{comm}}^\top$ is the gossip matrix of the communication graph
4: Agree on a common seed
5: $x_0^{(i)} = y_0^{(i)} = v_0^{(i)} = z_0^{(i)} = 0^{(n+nm)\times d}$ // *Initialization*
6: **for** $t = 0$ to $K-1$ **do** // *Run for K iterations*
7: $\quad y_t^{(i)} = \frac{1}{1+\rho}\left(x_t^{(i)} + \rho v_t^{(i)}\right)$
8: $\quad$ Draw u_{comm} uniform in $[0,1]$, using the common seed.
9: $\quad$ **if** $u_{\text{comm}} < p_{\text{comm}}$ (communication update) **then**
10: $\quad\quad$ Broadcast $y_t^{(i)}$ and retrieve $y_t^{(\ell)}$ from neighbours.
11: $\quad\quad \delta_t^{(i)} = \frac{\eta}{p_{\text{comm}}} \sum_{\ell=1}^n W_{i\ell} \frac{y_t^{(\ell)}}{\sigma_\ell}$ // *Gossip Communication*
12: $\quad\quad z_{t+1}^{(i)} = v_{t+1}^{(i)} = (1-\rho) v_t^{(i)} + \rho y_t^{(i)} - \delta_t^{(i)}$
13: $\quad\quad x_{t+1}^{(i)} = y_t^{(i)} - \frac{\rho}{p_{\text{comm}}} \delta_t^{(i)}$
14: $\quad$ **else**
15: $\quad\quad$ Sample function j with probability p_{ij}/p_{comp} (which sum to 1).
16: $\quad\quad z_{t+1}^{(i)} = (1-\rho) v_t^{(i)} + \rho y_t^{(i)} - \frac{\eta \mu_{ij}^2}{p_{ij}} \left(\frac{y_t^{(i)}}{\sigma_i} - \frac{y_t^{(ij)}}{L_{ij}} \right)$
17: $\quad\quad z_{t+1}^{(ij)} = (1-\rho) v_t^{(ij)} + \rho y_t^{(ij)} - \frac{\eta \mu_{ij}^2}{p_{ij}} \left(\frac{y_t^{(ij)}}{L_{ij}} - \frac{y_t^{(i)}}{\sigma_i} \right)$
18: $\quad\quad v_{t+1}^{(ij)} = \text{prox}_{\eta \mu_{ij}^2 p_{ij}^{-1} \tilde{f}_{ij}^*} \left(z_{t+1}^{(ij)} \right)$ // *Virtual node update using f_{ij}*
19: $\quad\quad v_{t+1}^{(i)} = z_{t+1}^{(i)} + \sum_{j,(ij)\in b} (z_{t+1}^{(ij)} - v_{t+1}^{(ij)})$ // *Center node update*
20: $\quad\quad x_{t+1}^{(i)} = y_t^{(i)} + \frac{\rho}{p_{ij}} (v_{t+1}^{(i)} - (1-\rho) v_t^{(i)} - \rho y_t^{(i)})$
21: $\quad\quad x_{t+1}^{(ij)} = y_t^{(ij)} + \frac{\rho}{p_{ij}} (v_{t+1}^{(ij)} - (1-\rho) v_t^{(ij)} - \rho y_t^{(ij)})$
22: **return** $\theta_K^{(i)} = v_K^{(i)} / \sigma_i$ // *Return primal parameter*

Primal proximal updates. The proximal step of Line 10 is performed with the function $\tilde{f^*}_{ij} : x \to f_{ij}^*(x) - \frac{1}{2L_{ij}} \|x\|^2$ instead of f_{ij}. Yet, Moreau identity [Parikh and Boyd, 2014] provides a way to retrieve the proximal operator of f^* using the proximal operator of f, but this does not directly apply to $\tilde{f}_{ij}^*$, making its proximal update hard to compute when no analytical formula is available to compute $\tilde{f}_{ij}^*$. Fortunately, the proximal operator of $\tilde{f}_{ij}^*$ can be retrieved from the proximal operator of f_{ij}^*. Following the derivations from the conference paper [Hendrikx et al., 2019b], we now show how to implement Algorithm 4 in a primal-only way. More specifically, if we denote $\tilde{\eta}_{ij} = \eta \mu_{ij}^2 p_{ij}^{-1}$ then for any $x \in \mathbb{R}^{n+nm}$, we can also express the update only in terms of f_{ij}^*:

$$\begin{aligned}
\text{prox}_{\tilde{\eta}_{ij} \tilde{f}_{ij}^*}(x) &= \arg\min_v \frac{1}{2\tilde{\eta}_{ij}} \|v - x\|^2 + f_{ij}^*(v) - \frac{1}{2L_{ij}} \|v\|^2 \\
&= \arg\min_v \frac{1}{2} \left(\tilde{\eta}_{ij}^{-1} - L_{ij}^{-1} \right) \|v\|^2 - \tilde{\eta}_{ij}^{-1} v^\top x + f_{ij}^*(v)
\end{aligned}$$

$$= \arg\min_v \frac{1}{2\left(\tilde{\eta}_{ij}^{-1} - L_{ij}^{-1}\right)^{-1}} \|v - \left(1 - \tilde{\eta}_{ij} L_{ij}^{-1}\right)^{-1} x\|^2 + f_{ij}^*(v)$$
$$= \mathrm{prox}_{\left(\tilde{\eta}_{ij}^{-1} - L_{ij}^{-1}\right)^{-1} f_{ij}^*} \left(\left(1 - \tilde{\eta}_{ij} L_{ij}^{-1}\right)^{-1} x\right).$$

Then, we use the identity:

$$\mathrm{prox}_{(\eta f)^*}(x) = \eta \mathrm{prox}_{\eta^{-1} f^*}\left(\eta^{-1} x\right), \tag{3.6.7}$$

and the Moreau identity leads to:

$$\mathrm{prox}_{\eta f^*}(x) = x - \eta \mathrm{prox}_{\eta^{-1} f}\left(\eta^{-1} x\right). \tag{3.6.8}$$

This allows us to retrieve the proximal operator on $\tilde{f}_{ij}^*$ using only the proximal operator on f_{ij}:

$$\left(1 - \tilde{\eta}_{ij} L_{ij}^{-1}\right) \mathrm{prox}_{\tilde{\eta}_{ij} \tilde{f}_{ij}^*}(x) = x - \tilde{\eta}_{ij} \mathrm{prox}_{\left(\tilde{\eta}_{ij}^{-1} - L_{ij}^{-1}\right) f_{ij}}\left(\tilde{\eta}_{ij}^{-1} x\right). \tag{3.6.9}$$

Note that the previous calculations are valid as long as $\tilde{\eta}_{ij} L_{ij}^{-1} \leq 1$ for all virtual edges. Using the same values for μ_{ij}^2 as in Assumption 5, and using the fact that $\eta = \rho/\sigma_A = 2\rho/\alpha$, this condition writes $2\rho \leq p_{ij}$ for all virtual edges (ij). By definition of ρ we have $\rho \leq p_{ij}/\sqrt{2(1 + L_{ij}/\sigma_i)}$, so this constraint simply makes ρ smaller by a $\sqrt{2}$ factor in the worst case (and does not change anything as long as $L_{ij} > \sigma_i$ for all j). In the case of Algorithm 4, the update of Line 10 can be rewritten:

$$v_{t+1}^{(ij)} = \left(\tilde{\eta}_{ij}^{-1} - L_{ij}^{-1}\right)^{-1} \left[\tilde{\eta}_{ij}^{-1} z_{t+1}^{(ij)} - \mathrm{prox}_{(\tilde{\eta}_{ij}^{-1} - L_{ij}^{-1}) f_{ij}}\left(\tilde{\eta}_{ij}^{-1} z_{t+1}^{(ij)}\right)\right].$$

Communications. Communications in Algorithm 4 are abstracted by multiplication by the Matrix W_b for a given batch of coordinates b. Note that if b is a set of virtual edges then no communications in the network are required since W_b only requires information exchange between central nodes and their virtual nodes. The formulation of ADFS suggests that another communication round using the matrix $\widetilde{W}_b$ is required for the actual update. Yet, if b_t is such that $P_b^\dagger A^\dagger A P_b^\dagger$ is a diagonal matrix, then $\widetilde{W}_b W_b$ can be performed with only one round of communications. This is the case for example if b_t is a set of virtual edges (then no communication is required). If b_t is the set of all communication edges, then $\widetilde{W}_{b_t} = P_{b_t}^\dagger$ since $p_{\mathrm{comm}} P_{b_t}^\dagger$ is the identity on $\mathrm{Ker}(A)^\perp$ so no extra communication is required in this case either.

Linear case. For many standard machine learning problems, $f_{ij}(\theta) = \ell(X_{ij}^\top \theta)$ with $X_{ij} \in \mathbb{R}^d$. This implies that $f_{ij}^*(\theta) = +\infty$ whenever $\theta \notin \mathrm{Span}\,(X_{ij})$. Therefore, the proximal steps on the Fenchel conjugate only have support on X_{ij}, meaning that they are one-dimensional problems that can be solved in constant time using for example the Newton method when no analytical solution is available. Warm starts (initializing on the previous solution) can also be used for solving the local problems even faster so that in the end, a one-dimensional proximal update is only a constant time slower than a gradient update. Note that this also allows to store parameters v_t and y_t as scalar coefficients for virtual nodes, thus greatly reducing the memory footprint of ADFS, and speeding up the updates when X_{ij} are sparse. Finally, the projectors are equal to $P_{ij} = X_{ij} X_{ij}^\top / \|X_{ij}\|^2$ in this case which, as we will see, implies that $\kappa_{\mathrm{comm}} = \kappa_b$ when $\sigma_i = \sigma$ for all i. Yet, the convex combinations of lines 7 and

12 still involve dense vectors. To alleviate this cost, one can instead use an *efficient* version of APCG [Lee and Sidford, 2013, Lin et al., 2015b, Fercoq and Richtárik, 2015].

Virtual updates. If b is a block of virtual edges, then the update to v_{t+1} has support on $\mathrm{Ker}(f_{ij})^\perp$. Indeed, recall that Ae_{ij} has a projector part, and so $\eta W_b \Sigma^\dagger y_t$ is independent of the value of $y_t^{(i)}$ and $y_t^{(ij)}$ on $\mathrm{Ker}(f_{ij})$. This facilitates implementation in the linear case.

Unbalanced local datasets. We assume that all local datasets are of fixed size m in order to ease reading. Yet, the impact of the value of m on Algorithm 4 is indirect, and unbalanced datasets can be handled without any change.

Natural Strong Convexity. ADFS is derived when strong convexity is obtained through L2 regularization. It is possible to generalize this to arbitrary strongly convex functions ω_i by simply replacing $\frac{1}{2\sigma_i}\|\cdot\|^2$ by ω_i^*, and performing the same derivations. Yet, we chose to focus on the L2 regularization case to ease reading of the paper.

3.6.4. Non-smooth setting. The accelerated proximal coordinate gradient algorithm can be applied to the problem of Equation (3.6.4) even if the function q_A is not strongly convex on $\mathrm{Ker}(A)^\perp$. This is for instance the case when the functions f_{ij} are not smooth so that $\Sigma^\dagger$ has diagonal blocks equal to 0 and therefore $\mathrm{Ker}(A^\top\Sigma^\dagger A) \not\subset \mathrm{Ker}(A)$ so $\sigma_A = 0$. In this case, the choice of coefficients from Corollary 7 leads to NS-ADFS, that provides error guarantees when primal functions f_{ij} are not smooth. More formally, with $F^* : x \to \sum_{i=1}^n \left[\sum_{j=1}^m f_{ij}^*\left(x^{(ij)}\right) + \frac{1}{2\sigma_i}\|x^{(i)}\|^2\right]$, we have:

THEOREM 15. *If the functions f_{ij} are non-smooth then NS-ADFS guarantees:*

$$\mathbb{E}[F^*(x_t)] - F^*(\theta^\star) \leq \frac{2}{t^2}\left[\frac{S^2}{\lambda_{\min}^+(A^\top A)}r_t^2 + \frac{6}{p_{\min}^2}\left[F^*(x_0) - F^*(\theta^\star)\right]\right],$$

with $S^2 = \max_b \lambda_{\max}(A^\dagger A P_b^\dagger A^\top \Sigma^\dagger A P_b^\dagger A^\dagger A)$, $r_t^2 = \|v_0 - \theta^\star\|^2 - \|v_t - \theta^\star\|^2$ and $p_{\min}$ is taken over virtual edges.

The guarantees provided by Theorem 15 are weaker than in the smooth setting. In particular, we lose linear convergence and get the classical accelerated sublinear $O(1/t^2)$ rate. We also lose the bound on the primal parameters— recovering primal guarantees is beyond the scope of this work. Note that the extra $\lambda_{\min}^+(A^\top A)$ term comes from the fact that Theorem 15 is formulated with primal parameter sequences $x_t = A\widetilde{x}_t$. Also note that $\alpha_t = O\left(t^{-1}\right)$.

3.6.5. Performances and Parameters Choice in the Homogeneous Setting. We now prove the time to convergence of ADFS presented in Table 1, and detail the conditions under which it holds. Indeed, Section 3.6.2 presents ADFS in full generality but the different parameters have to be chosen carefully to reach optimal speed. In particular, we have to choose the coefficients μ to make sure that the graph augmentation trick does not cause the smallest positive eigenvalue of $A^\top\Sigma^\dagger A$ to shrink too much, which is done by Lemma 9.

ASSUMPTION 5 (Parameters choice). *For arbitrary $\mu_{k\ell}$ and for all communication edges, we denote $L = A_{\mathrm{comm}}A_{\mathrm{comm}}^\top \in \mathbb{R}^{n\times n}$ the Laplacian of the communication graph. Let D_M and $\widetilde{D}_M$ be the diagonal matrices such that $(D_M)_{ii} = \sigma_i + \lambda_{\max}\left(\sum_{j=1}^m L_{ij}P_{ij}\right)$ and $(\widetilde{D}_M)_{ii} = \sigma_i + 2\lambda_{\max}\left(\sum_{j=1}^m L_{ij}P_{ij}\right)$. The local condition number of node i is $\kappa_i = (D_M)_{ii}/\sigma_i$, and we*

choose the weights of virtual edges as $\mu_{ij}^2 = \alpha L_{ij}$, *with* $\alpha = 2\lambda^+_{\min}(A_{\text{comm}}^\top \widetilde{D}_M^{-1} A_{\text{comm}})$, *and their probabilities as* $p_{ij} = p_{\text{comp}}(1 + L_{ij}\sigma_i^{-1})^{\frac{1}{2}}/S_i$ *with* $S_i = \sum_{j=1}^m (1 + L_{ij}\sigma_i^{-1})^{\frac{1}{2}}$.

This choice of parameters allows to tightly bound $\lambda^+_{\min}(A^\top \Sigma^\dagger A)$, which defines the rate of convergence of ADFS.

LEMMA 9. *If Assumption 5 holds, then for any* $x \in \mathbb{R}^{E+nm}$ *we have that* $\|x\|^2_{A^\top \Sigma^\dagger A} \geq \lambda^+_{\min}(A_{\text{comm}}^\top \widetilde{D}_M^{-1} A_{\text{comm}}) \|x\|^2_{A^\dagger A}$. *In particular,* $\sigma_A \geq \alpha/2$.

The proof studies the Schur complement of $\Sigma^{-\frac{1}{2}} A A^\top \Sigma^{-\frac{1}{2}}$. This yields a characterization of the eigenvalues of $A^\top \Sigma^\dagger A$ in terms of a determinant equation of the form $\det(L_{\text{comm}} - \Delta_\lambda) = 0$, with Δ_λ a block-diagonal matrix that depends on λ and $L_{\text{comm}} = A_{\text{comm}} A_{\text{comm}}^\top$, where $A_{\text{comm}} \in \mathbb{R}^{nd \times Ed}$ is the restriction of A to communication nodes and edges. Lemma 10 gives necessary conditions for an x to be in $\text{Ker}(L_{\text{comm}} - \Delta_\lambda)$, and we thus deduce bounds on the smallest eigenvalue of $A^\top \Sigma^\dagger A$ from upper bounds on Δ_λ. Note that the proof is simpler than in the conference paper [Hendrikx et al., 2019b], and the choices in Assumption 5 allow for a tighter bound. We start by a simple Lemma and then proceed to the actual proof.

LEMMA 10. *Let* $U, V \in \mathbb{R}^d$ *be two symmetric positive semi-definite matrices. Let* $x \in \text{Ker}(U - V)$, *that can be decomposed into* $x = x_+ + x_\perp$, *with* $x_\perp \in \text{Ker}(U)$ *and* $x_+ \in \text{Ker}(U)^\perp$. *Then,* $x_+^\top U x_+ \leq x_+^\top V x_+$, *and if* $x_+ = 0$ *then* $x_\perp \in \text{Ker}(V)$.

PROOF. Let $x \in \text{Ker}(U - V)$. We write:

$$x_+^\top U x_+ = x^\top U x = x^\top V x = x_+^\top V x_+ + 2x_\perp^\top V x_+ + x_\perp^\top V x_\perp.$$

Besides, $x_\perp^\top (U - V) x = 0$, and so $x_\perp^\top V x_+ = -x_\perp^\top V x_\perp \leq 0$. Therefore,

$$x_+^\top U x_+ = x_+^\top V x_+ + x_\perp^\top V x_+ \leq x_+^\top V x_+.$$

Finally, if $x_+ = 0$ then $x_\perp^\top V x_\perp = -x_\perp^\top V x_+ = 0$, and so $x_\perp \in \text{Ker}(V)$. □

PROOF OF LEMMA 9. For any rectangular matrix Q, all non-zero singular values of the matrix $Q^\top Q$ are also non-zero singular values of the matrix $QQ^\top$, so we can analyze the spectrum of the matrix $\tilde{L} = \Sigma^{-1/2} A A^\top \Sigma^{-1/2}$ instead of the spectrum of $A^\top \Sigma^{-1} A$. Recall that A writes:

$$A = \begin{pmatrix} A_{\text{comm}} \otimes I_d & D_\mu \\ 0 & -D_\mu^{\text{diag}} \end{pmatrix} \tag{3.6.10}$$

Then, if we denote L_{comm} the Laplacian matrix of the original true graph, the rescaled Laplacian matrix of the augmented graph writes:

$$\tilde{L} = \Sigma^{-1/2} \begin{pmatrix} L_{\text{comm}} \otimes I_d + D_\mu D_\mu^\top & -D^\mu D_\mu^{\text{diag}} \\ -D_\mu^{\text{diag}} D_\mu^\top & (D_\mu^{\text{diag}})^2 \end{pmatrix} \Sigma^{-1/2}. \tag{3.6.11}$$

We define $P_\perp = \text{Diag}(P_{11}, \cdots, P_{nm}) \in \mathbb{R}^{nmd \times nmd}$. If we split Σ into two diagonal blocks $\Sigma_{\text{comm}} = \text{diag}(\sigma_1, \cdots, \sigma_n) \otimes I_d$ (for the communication nodes), and Σ_{comp} (for the computation nodes) and apply the block determinant formula, we obtain:

$$\begin{aligned} \det(\tilde{L} - \lambda I_{n(m+1)d}) &= \det(\Sigma_{\text{comp}}^{-\frac{1}{2}} (D_\mu^{\text{diag}})^2 \Sigma_{\text{comp}}^{-\frac{1}{2}} - \lambda I_{nmd}) \\ &\times \det(\Sigma_{\text{comm}}^{-\frac{1}{2}} [(L_{\text{comm}} \otimes I_d) + D_\mu D_\mu^\top - \lambda \Sigma_{\text{comm}} - \end{aligned}$$

$$D_\mu D_\mu^{\text{diag}} \Sigma_{\text{comp}}^{-\frac{1}{2}} \left(\Sigma_{\text{comp}}^{-\frac{1}{2}} (D_\mu^{\text{diag}})^2 \Sigma_{\text{comp}}^{-\frac{1}{2}} - \lambda I_{nmd} \right)^\dagger \Sigma_{\text{comp}}^{-\frac{1}{2}} D_\mu^{\text{diag}} D_\mu^\top] \Sigma_{\text{comm}}^{-\frac{1}{2}}).$$

Now, we use that $\mu_{ij}^2 = \alpha L_{ij}$ for some $\alpha > 0$, and note that:

$$(\Sigma_{\text{comp}}^{-\frac{1}{2}} (D_\mu^{\text{diag}})^2 \Sigma_{\text{comp}}^{-\frac{1}{2}})_{ij} = (D_\mu^{\text{diag}} \Sigma_{\text{comp}}^{-1} D_\mu^{\text{diag}})_{ij} = \mu_{ij}^2 P_{ij} / L_{ij} = \alpha P_{ij}.$$

This can be used along with the fact that $D_\mu D_\mu^\top = D_\mu P_\perp D_\mu^\top$ to write:

$$\begin{aligned} \det(\tilde{L} - \lambda I_{n(m+1)d}) &= \det(\alpha P_\perp - \lambda I_{nmd}) \times \\ &\quad \det(L_{\text{comm}} \otimes I_d - \lambda \Sigma_{\text{comm}} - \alpha D_\mu P_\perp (\alpha P_\perp - \lambda I_{nmd})^\dagger D_\mu^\top). \end{aligned}$$

Note that since $P_\perp$ is a projector, $P_\perp (\alpha P_\perp - \lambda I_{nmd})^\dagger = ((\alpha P_\perp - \lambda I_{nmd}) P_\perp)^\dagger = (\alpha - \lambda)^{-1} P_\perp$. Therefore, the non-zero eigenvalues of $A^\top \Sigma^{-1} A$ are the λ that satisfy the following equation:

$$0 = \det(\alpha P_\perp - \lambda I_{nmd}) \det \left(L_{\text{comm}} \otimes I_d - \lambda \left(\Sigma_{\text{comm}} + \frac{1}{\alpha - \lambda} D_\mu D_\mu^\top \right) \right). \tag{3.6.12}$$

We now consider $0 < \lambda \leq \alpha/2$, so that: $\Sigma_{\text{comm}} + \frac{1}{\alpha - \lambda} D_\mu D_\mu^\top \preccurlyeq \tilde{D}_M \otimes I_d$, with

$$(\tilde{D}_M)_{ii} = \lambda_{\max} \left(\sigma_i I_d + \frac{2}{\alpha} \sum_{j=1}^m \mu_{ij}^2 P_{ij} \right) = \sigma_i + 2\lambda_{\max} \left(\sum_{j=1}^m L_{ij} P_{ij} \right).$$

Therefore, for any $y \in \text{Ker}(L_{\text{comm}} \otimes I_d)^\perp$ such that $y \neq 0$,

$$y^\top (L_{\text{comm}} \otimes I_d - \Delta_\lambda) y > y^\top ((L_{\text{comm}} - \lambda \tilde{D}_M) \otimes I_d) y.$$

In particular we have that if $0 < \lambda < \lambda_{\min}^+ (\tilde{D}_M^{-\frac{1}{2}} L_{\text{comm}} \tilde{D}_M^{-\frac{1}{2}})$ then

$$y^\top (L_{\text{comm}} \otimes I_d - \Delta_\lambda) y > 0. \tag{3.6.13}$$

Let $x \in \text{Ker}(L_{\text{comm}} \otimes I_d - \Delta_\lambda)$ then Lemma 10 tells us that $x = x_+ + x_\perp$ with $x^+ \in \text{Ker}(L_{\text{comm}} \otimes I_d)^\perp$ and $x_+^\top (L_{\text{comm}} \otimes I_d - \Delta_\lambda) x_+ \leq 0$. If $x_+ \neq 0$ then this contradicts Equation (3.6.13) so $x_+ = 0$, meaning that $x \in \text{Ker}(\Delta_\lambda)$ using the second part of Lemma 10, and so $x = 0$.

Therefore, if λ is such that $0 < \lambda < \min(\lambda_{\min}^+ (\tilde{D}_M^{-\frac{1}{2}} L_{\text{comm}} \tilde{D}_M^{-\frac{1}{2}}), \alpha/2)$ then by using the eigenvalue characterization given by Equation (3.6.12), λ is not an eigenvalue of $A^\top \Sigma^{-1} A$ since $\text{Ker}(L_{\text{comm}} - \Delta_\lambda) = \{0\}$, so in particular $\lambda_{\min}^+ (A^\top \Sigma^{-1} A) \geq \min(\tilde{D}_M^{-\frac{1}{2}} L_{\text{comm}} \tilde{D}_M^{-\frac{1}{2}}, \alpha/2)$.

Besides, $\lambda_{\min}^+ (\tilde{D}_M^{-\frac{1}{2}} L_{\text{comm}} \tilde{D}_M^{-\frac{1}{2}}) = \lambda_{\min}^+ (A_{\text{comm}}^\top \tilde{D}_M^{-1} A_{\text{comm}})$ and $A_{\text{comm}}^\top \tilde{D}_M^{-1} A_{\text{comm}}$ is independent of α since α only affects the μ_{ij} weights when (i, j) is a computation edge, and so we can choose $\alpha = 2\lambda_{\min}^+ (A_{\text{comm}}^\top \tilde{D}_M^{-1} A_{\text{comm}})$, so that $\lambda_{\min}^+ (A^\top \Sigma^{-1} A) \geq \lambda_{\min}^+ (A_{\text{comm}}^\top \tilde{D}_M^{-1} A_{\text{comm}})$. Finally, $\text{Ker}(A^\top \Sigma^{-1} A) = \text{Ker}(A)$ and so

$$\|x\|_{A^\top \Sigma^{-1} A}^2 \geq \lambda_{\min}^+ (A^\top \Sigma^{-1} A) \|x\|_{A^\dagger A}^2 \geq \lambda_{\min}^+ (A_{\text{comm}}^\top \tilde{D}_M^{-1} A_{\text{comm}}) \|x\|_{A^\dagger A}^2,$$

which finishes the proof. □

We now study parameter ρ more in details, which is defined in Equation (3.6.5) by bounding the spectrum of a matrix that depends on the block of coordinates chosen. The spectral properties of this matrix heavily depend on whether the block contains actual communication edges or virtual edges. One can trade p_{comp} for p_{comm} so that the bound is the same for both kind of edges. This amounts to tuning the ratio between communications and

computations. We first make some assumptions on the sampling performed, and then detail the communication and computation rate under this sampling.

ASSUMPTION 6 (Synchronous sampling). *The sampling of edges is such that:*

- *With probability p_{comm}, $b_t = b_{\text{comm}}$, the set of all communication edges. This corresponds to communicating over all edges of the network, which comes down to a multiplication by the gossip matrix L.*
- *With probability $p_{\text{comp}} = 1 - p_{\text{comm}}$, a computation step is performed. In this case, $b_t = \{(i, j_t(i)),\ i \in \{1, ..., n\}\}$, where $j_t(i) = j$ with probability p_{ij}. This corresponds to each node sampling exactly one virtual edge.*

This synchronous sampling defines the blocks of coordinates b_t that are picked by Algorithm 4. It is then possible to compute ρ, the rate of convergence of ADFS, depending on the frequency of communication p_{comm}.

LEMMA 11. *We denote $\kappa_s = \max_i \kappa_i$ and γ the spectral gap of the Laplacian of the communication graph $L_{\text{comm}} = A_{\text{comm}} A_{\text{comm}}^\top$. Under the synchronous sampling of Assumption 6, the convergence rate of ADFS is such that*

$$\rho^2 = \min\left(\frac{\gamma}{\kappa_{\text{comm}}} p_{\text{comm}}^2, \frac{p_{\text{comp}}^2}{2(m + \sqrt{m\kappa_s})^2}\right), \text{ with}$$
$$\kappa_{\text{comm}} = \frac{\lambda_{\max}(A_{\text{comm}}^\top \Sigma_{\text{comm}}^{-1} A_{\text{comm}}) \ / \ \lambda_{\max}(A_{\text{comm}}^\top A_{\text{comm}})}{\lambda_{\min}^+(A_{\text{comm}}^\top \widetilde{D}_M^{-1} A_{\text{comm}}) \ / \ \lambda_{\min}^+(A_{\text{comm}}^\top A_{\text{comm}})}.$$

If $\sigma_k = \sigma$ for all k (homogeneous case) then $\kappa_{\text{comm}} = \max_i (\widetilde{D}_M)_{ii}/\sigma$. If $f_{ij}(\theta) = g(X_{ij}^\top \theta)$ and g is L_g-smooth then $(\widetilde{D}_M)_{ii} = \sigma_i + 2L_g \lambda_{\max}\left(\sum_{j=1}^m X_{ij} X_{ij}^\top\right)$. Therefore, κ_{comm} is of order κ_b rather than κ_s, and we recover the expected communication complexity for decentralized algorithms.

PROOF OF LEMMA 11. The proof is split into three parts. We first bound the value of $\lambda_{\max}((A^\dagger A)^\top P_b^\dagger A^\top \Sigma^{-1} A P_b^\dagger A^\dagger A)$ depending on whether b is a communication or a computation block, and then we give a bound on the rate ρ.

Communication blocks. Under the sampling of Assumption 6, all coordinates have the same probability p_e of being selected at each step. In this case, $P_b^\dagger = \frac{1}{p_b} U_b$ where U_b is the projector on communication edges that are in b (all the communication edges for Assumption 6. We denote $V_b \in \mathbb{R}^{(m+1)nd \times (m+1)nd}$ the projector on $\{i,\ \exists j / (i,j) \in b\}$, the set of nodes for which one of their vertices is updated, and write:

$$P_b^\dagger A^\top \Sigma^{-1} A P_b^\dagger = p_e^{-2} U_b A^\top \Sigma^{-1} A U_b = p_e^{-2} U_b A^\top V_b \Sigma^{-1} V_b A U_b.$$

In this case, the Laplacian L_b of the subgraph defined by the edges in b is such that $L_b \otimes I_d = A U_b A^\top$, and we use that $\lambda_{\max}(A U_b A^\top) = \lambda_{\max}(U_b A^\top A U_b)$ to write:

$$\lambda_{\max}((A^\dagger A)^\top P_b^\dagger A^\top \Sigma^{-1} A P_b^\dagger A^\dagger A) \leq \frac{\lambda_{\max}(L_b)}{\sigma_{\min} p_e^2}. \tag{3.6.14}$$

In particular, Equation (3.6.14) allows to consider dynamically changing graphs for which we know that all edges have the same probability of appearing at each step and for which we can

bound the Laplacian matrix of any subgraph. This allows to consider a complete underlying communication graph while taking advantage of communications on subgraphs only. If we take $p_e = p_{\text{comm}}$ (i.e. all communication edges are sampled at each communication step) then this becomes:

$$\lambda_{\max}((A^\dagger A)^\top P_b^\dagger A^\top \Sigma^{-1} A P_b^\dagger A^\dagger A) \leq \frac{\lambda_{\max}(L)}{\sigma_{\min} p_{\text{comm}}^2},$$

with L the Laplacian matrix of the original communication graph.

Computation blocks. We start with the case in which each node only samples the coordinate associated with one virtual edge. In this case, we take $\lambda \in \mathbb{R}^{E+nmd}$ and write:

$$\begin{aligned}
&(e_b \otimes \lambda)^\top P_b^\dagger A^\top \Sigma^{-1} A P_b^\dagger (e_b \otimes \lambda) \\
&= \sum_{i=1}^n \sum_{j,(i,j)\in b} e_{ij}^\top A^\top \sum_{u\in V} \Sigma_{uu}^{-1} e_u e_u^\top \sum_{i'=1}^n \sum_{j',(i',j')\in b} A e_{i'j'} \times \lambda_{ij}^\top P_{ij} P_{i'j'} \lambda_{i'j'} \\
&= \sum_{i=1}^n \sum_{j,(i,j)\in b} \sum_{i'=1}^n \sum_{j',(i',j')\in b} \frac{\mu_{ij}}{p_{ij}} \sum_{u\in V} \Sigma_{uu}^{-1} (e_i - e_j)^\top e_u e_u^\top (e'_i - e'_j) \lambda_{ij}^\top P_{ij} P_{i'j'} \lambda_{i'j'} \\
&= \sum_{i=1}^n \sum_{j,(i,j)\in b} \frac{\mu_{ij}^2 (\sigma_i^{-1} + f_{ij}^{-1})}{p_{ij}^2} \times \|\lambda_{ij}\|_{P_{ij}}^2.
\end{aligned}$$

We deduce that if only one coordinate is sampled per node then we have:

$$\lambda_{\max}(P_b^\dagger A^\top \Sigma^{-1} A P_b^\dagger) \leq \max_{i,j} \frac{\mu_{ij}^2 (\sigma_i^{-1} + f_{ij}^{-1})}{p_{ij}^2}.$$

Rate of convergence. The rate of convergence of Synch-ADFS depends on:

$$\rho^2 = \min_b \frac{\sigma_A}{\lambda_{\max}((A^\dagger A)^\top P_b^\dagger M P_b^\dagger A^\dagger A)} = \min_b \frac{\lambda_{\min}^+(A_{\text{comm}}^\top \tilde{D}_M^{-1} A_{\text{comm}})}{\lambda_{\max}((A^\dagger A)^\top P_b^\dagger A^\top \Sigma^{-1} A P_b^\dagger A^\dagger A)},$$

where $(\tilde{D}_M)_{ii} = \sigma_i + 2\lambda_{\max}\left(\sum_{j=1}^m f_{ij} P_{ij}\right)$. If we take b to be the set of all communication edges, then we obtain:

$$\begin{aligned}
\rho_{\text{comm}}^2 &= \frac{\lambda_{\min}^+(A_{\text{comm}}^\top \tilde{D}_M^{-1} A_{\text{comm}})}{\lambda_{\max}(A_{\text{comm}}^\top \Sigma_{\text{comm}}^{-1} A_{\text{comm}})} p_{\text{comm}}^2 = \frac{\gamma p_{\text{comm}}^2}{\kappa_{\text{comm}}}, \\
\rho_{\text{comp}}^2 &= \min_{ij} \frac{p_{ij}^2 \sigma_A}{\mu_{ij}^2 (\sigma_i^{-1} + L_{ij}^{-1})} = \min_{ij} \frac{p_{ij}^2}{2(1 + L_{ij}\sigma_i^{-1})},
\end{aligned} \tag{3.6.15}$$

where we used in the second equation that $\mu_{ij}^2 = \alpha L_{ij}$ and $\alpha = 2\sigma_A$. The constraint on ρ_{comp}^2 is that all p_{ij} are normalized separately, i.e. $\sum_j p_{ij} = 1$ for each node i. Indeed, exactly one sample per node is chosen at each step, and so:

$$\rho_{\text{comp}}^2 = \frac{p_{\text{comp}}^2}{2S_i^2} \geq \frac{p_{\text{comp}}^2}{2S_{\max}^2}. \tag{3.6.16}$$

We finally use the concavity of the square root with Jensen inequality to get:

$$S_i = \sum_{j=1}^{m} \sqrt{1 + L_{ij}\sigma_i^{-1}} \leq m \sqrt{\sum_{j=1}^{m} \frac{1}{m}(1 + L_{ij}\sigma_i^{-1})} = m\sqrt{1 + \sum_{j=1}^{m} \frac{L_{ij}}{m\sigma_i}},$$

which yields $S_{\max}^2 \leq m^2 + m\kappa_s$, or $S_{\max} \leq m + \sqrt{m\kappa_s}$. □

Now that we have specified the rate of ADFS (improvement per iteration), the only step left is to tune p_{comm} to minimize time needed to reach a given precision ε. Theorem 16 gives a choice of p_{comm} that achieves optimal rates. Note that similar derivations can be performed in the non-smooth case, but we do not detail them here due to lack of space.

THEOREM 16. *If* $p_{\text{comm}} = \left(1 + \sqrt{\frac{2\gamma}{\kappa_{\text{comm}}}}(m + \sqrt{m\kappa_s})\right)^{-1}$, *then running Algorithm 4 for* $K = \rho^{-1}\log(\varepsilon^{-1})$ *iterations guarantees* $\mathbb{E}\left[\|\theta_K - \theta^\star\|^2\right] \leq C_0\varepsilon$, *and takes time* $T(K)$, *with* $T(K)$ *such that:*

$$\mathbb{E}[T(K)] \leq \left(\sqrt{2}(m + \sqrt{m\kappa_s}) + \tau\sqrt{\frac{\kappa_{\text{comm}}}{\gamma}}\right)\log\left(\frac{1}{\varepsilon}\right).$$

PROOF. The execution time of the algorithm $T(K)$ verifies:

$$\mathbb{E}[T(K)] = (p_{\text{comp}} + \tau p_{\text{comm}})\, K \tag{3.6.17}$$

We know from Theorem 14 that using Algorithm 4 for $K_\varepsilon = \log(1/\varepsilon)\rho^{-1}$ steps guarantees to reach an error smaller than $C_0\varepsilon$. Rewriting this in terms of ρ_{comm} and ρ_{comp} (defined in (3.6.15)), we obtain:

$$\frac{\mathbb{E}[T(K_\varepsilon)]}{\log(\varepsilon^{-1})} = \max\left(T_1(p_{\text{comm}}), T_2(p_{\text{comm}})\right), \text{ with} \tag{3.6.18}$$

$$T_1(p_{\text{comm}}) = \rho_{\text{comm}}^{-1}(p_{\text{comp}} + \tau p_{\text{comm}}) = C_{\text{comm}}\left(\tau - 1 + \frac{1}{p_{\text{comm}}}\right), \tag{3.6.19}$$

$$T_2(p_{\text{comm}}) = \rho_{\text{comp}}^{-1}(p_{\text{comp}} + \tau p_{\text{comm}}) = C_{\text{comp}}\left(1 + \tau\frac{p_{\text{comm}}}{1 - p_{\text{comm}}}\right), \tag{3.6.20}$$

where $C_{\text{comm}}^2 = \frac{\kappa_{\text{comm}}}{\gamma}$ and $C_{\text{comp}}^2 = 2S_{\max}^2$ which are independent of p_{comp} and p_{comm}. T_1 is a continuous decreasing function of p_{comm} with $T_1 \to \infty$ when $p_{\text{comm}} \to 0$. Similarly, T_2 is a continuous increasing function of p_{comm} such that $T_2 \to \infty$ when $p_{\text{comm}} \to 1$. Therefore, the best upper bound on the execution time is given by taking $p_{\text{comm}} = p^*$ where p^* is such that $T_1(p^*) = T_2(p^*)$ and so $\rho_{\text{comm}}(p^*) = \rho_{\text{comp}}(p^*)$. More specifically, p^* is the solution of

$$p_{\text{comp}}^2 = p_{\text{comm}}^2\frac{C_{\text{comp}}}{C_{\text{comm}}} = (1 - p_{\text{comm}})^2, \text{ so } p^* = \left(1 + \frac{C_{\text{comp}}}{C_{\text{comm}}}\right)^{-\frac{1}{2}}. \tag{3.6.21}$$

Plugging it back into Equation (3.6.19) and using that $T_1(p^*) = T_2(p^*)$, we get $\frac{\mathbb{E}[T(K_\varepsilon)]}{\log(\varepsilon^{-1})} = C_{\text{comp}} + C_{\text{comm}}\tau$ which completes our proof. □

For generalized linear models with homogeneous regularization, we already saw that $\kappa_{\text{comm}} = O(\kappa_b)$, and thus ADFS is directly optimal. In heterogeneous cases, $\kappa_{\text{comm}} \leq \max_i \widetilde{D}_M / \min_i \sigma_i$, but tighter bounds can be derived by capturing the interplay between the regularity of the local functions and the topology of the communication graph. It is

for instance the case for the functions used to derive the lower bound from Corollary 5. Indeed, these functions do not have that $\sigma_k = \sigma_\ell$ for all k, ℓ, so we cannot directly say that $\kappa_{\text{comm}} = \kappa_b$ in this case. Fortunately, it is possible to exploit the structure of the graph and of $\widetilde{D}_M$ and $\Sigma_{\text{comm}}^{\dagger}$ to derive that $\kappa_{\text{comm}} = O(\kappa_b)$ anyway, as shown in the following corollary.

COROLLARY 8 (Tightness of the lower bound). *The worst-case functions and graph used in Corollary 5 are such that $\kappa_{\text{comm}} \leq 12\kappa_b$, so in particular the rate given by Theorem 16 matches the rate of the lower bound from Corollary 5.*

PROOF. We consider the same functions f_{ij} as in Theorem 12. In the decentralized case, the graph considered is a line graph. In order to simplify the exposition, we assume without loss of generality that n is odd and we choose parameter Δ (for the set Q_Δ^c) such that $\Delta = (n-1)/2$ instead of the diameter. One can verify that as long as $L \geq \sigma/3$, these worst-case functions verify:

$$\Sigma_{\text{comm}} \succcurlyeq \frac{\sigma}{3n} D_n \text{ and } \tilde{D}_M \preccurlyeq \frac{L}{n} D_n, \text{ with } D_n = \text{Diag}(n, 2, \cdots, 2) \in \mathbb{R}^n.$$

From there, using the structure of the line graph and of D_n, we obtain that for $n \geq 2$, $A_{\text{comm}}^\top D_n^{-1} A_{\text{comm}} = \frac{1}{2} A_{\text{comm}}^\top A_{\text{comm}} - \frac{\mu_{\text{comm}}^2}{2}\left(1 - \frac{2}{n}\right) e_{12} e_{12}^\top$, and so:

$$\lambda_{\max}(A_{\text{comm}}^\top \Sigma_{\text{comm}}^{-1} A_{\text{comm}}) \leq \frac{3n\lambda_{\max}(A_{\text{comm}}^\top A_{\text{comm}})}{2\sigma}. \tag{3.6.22}$$

For the second part, we consider the matrix $A_{\text{comm}}^\top \text{Diag}(\alpha, 1, \cdots, 1) A_{\text{comm}}$ and study $P_{n,\alpha}$, its characteristic polynomial, which can be obtained from the recursion:

$$P_{n+1,\alpha}(\lambda) = (1 + \alpha - \lambda) P_{n,1}(\lambda) - P_{n-1,1}(\lambda). \tag{3.6.23}$$

Writing λ as $\lambda = 2(1 - \cos(\vartheta))$, we obtain for $\alpha = 1$, $P_{n,1}(\lambda) = \sin(n\vartheta)/\sin\vartheta$ and we recover the standard eigenvalues λ of the line graph. For $\alpha = 0$, the zeros of $P_{n,0}$ are obtained for ϑ that satisfy $\sin(n\vartheta) = \sin((n-1)\vartheta)$. We solve for ϑ_k explicitly, and the smallest eigenvalue is obtained for $\vartheta_0 = \pi/(2n-1)$. Using that $\tilde{D}_M^{-1} = (n/2L)\text{Diag}(2/n, 1, \cdots, 1)$, simple algebra then leads to:

$$\lambda_{\min}^+(A_{\text{comm}}^\top \tilde{D}_M^{-1} A_{\text{comm}}) \geq \frac{2n}{2L}(1 - \cos\vartheta_0) \geq \frac{2n}{8L}(1 - \cos\frac{\pi}{n}) = \frac{n}{8L}\lambda_{\min}^+(A_{\text{comm}}^\top A_{\text{comm}}).$$

This allows to evaluate κ_{comm} and obtain the final result. □

3.7. Local synchrony and the randomized gossip model

In this section, we introduce a *locally synchronous* version of ADFS, in which communications are performed in the same way as in Chapter 2. The main difference with the synchronous version presented in Algorithm 5 is that from Algorithm 4, the communication and computation blocks b are of size 1, and thus restricted to one edge only.

ASSUMPTION 7 (Randomized sampling). *The sampling of edges is such that:*

- *Let $k, \ell \in \{1, \ldots, n\}$. With probability $p_{k\ell}$, $b_t = \{(k, \ell)\}$, the communication edge between nodes k and ℓ. Note that $\sum_{(k,\ell)\in E} p_{k\ell} = p_{\text{comm}}$.*

- *Let $i \in \{1, \dots, n\}$ and $j \in \{1, \dots, m\}$. With probability p_{ij}, $b_t = \{(i, j)\}$ the computation edge between node i and its j-th virtual node. This means that node i performs a local computation computation step using function f_{ij}. Note that $\sum_{i=1}^{n} \sum_{j=1}^{m} p_{ij} = p_{\text{comp}}$.*

3.7.1. Distributed Execution and Synchronization Time. Theorem 14 gives bounds on the expected error after a given number of iterations. To assess the actual speed of the algorithm, it is still required to know how long executing a given number of iterations takes. This is easy with Algorithm 5, since it is a synchronous algorithm in which all nodes iteratively perform local updates or communication rounds. In this case, executing n_{comp} computing rounds and n_{comm} communication rounds simply takes time $n_{\text{comp}} + \tau n_{\text{comm}}$. Yet, we study in this section a variant of ADFS that relies on randomized pairwise communications, so it is necessary to sample a *schedule*, *i.e.*, a random sequence of edges from the augmented graph, and evaluate how fast this schedule can be executed. Note that the execution time crucially depends on how many edges can be updated in parallel, which itself depends on the graph and on the random schedule sampled.

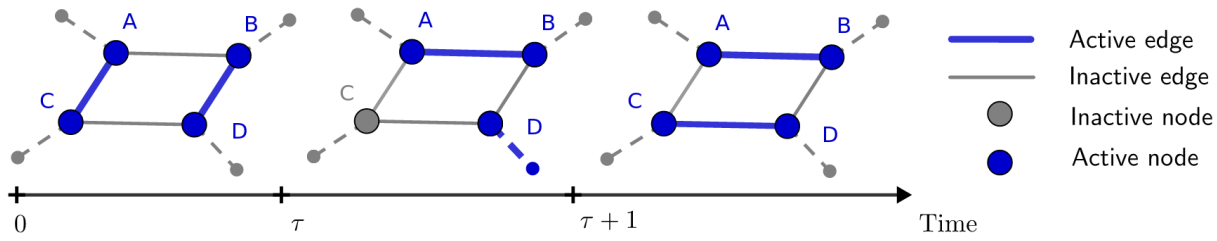


FIGURE 2. Illustration of parallel execution and local synchrony. Nodes from a toy graph execute the schedule $[(A, C), (B, D), (A, B), (D), (C, D)]$, where (D) means that node D performs a local update. Each node needs to execute its updates in the partial order defined by the schedule. In particular, node C has to perform update (A, C) and then update (C, D), so it is idle between times τ and $\tau + 1$ because it needs to wait for node D to finish its local update before the communication update (C, D) can start. We assume $\tau > 1$ since the local update terminates before the communication update (A, B). Contrary to synchronous algorithms, no global notion of rounds exist and some nodes (such as node D) perform more updates than others.

Shared schedule. Even though they only actively take part in a small fraction of the updates, all nodes need to execute the same schedule to correctly implement ADFS with local pairwise communications. To generate this shared schedule, all nodes are given a seed and the sampling probabilities of all edges. This allows them to avoid deadlocks and to precisely know how many convex combinations to perform between v_t and y_t.

Execution time. The problem of bounding the probability that a random schedule of fixed length exceeds a given execution time can be cast in the framework of fork-join queuing networks with blocking [Zeng et al., 2018]. In particular, queuing theory [Baccelli et al., 1992] tells us that the average time per iteration exists for any fixed probability distribution over a given augmented graph. Unfortunately, existing quantitative results are not precise enough for our purpose so we generalize the method introduced by Hendrikx et al.

[2019a] to get a finer bound. While their result is valid when the only possible operation is communicating with a neighbor, we extend it to the case in which nodes can also perform local computations. For the rest of this paper, we denote p_{comm} the probability of performing a communication update and p_{comp} the probability of performing a local update. They are such that $p_{\text{comp}} + p_{\text{comm}} = 1$. We also define $p_{\text{comm}}^{\max} = n \max_k \sum_{\ell \in \mathcal{N}(k)} p_{k\ell}/2$, where neighbors are in the communication network only. When all nodes have the same probability to participate in an update, $p_{\text{comm}}^{\max} = p_{\text{comm}}$. Then, the following theorem holds (see proof in Appendix 3.B):

THEOREM 17. *Let $T(t)$ be the time needed for the system to execute a schedule of size t,* i.e.*, t iterations of Algorithm 4 under the randomized sampling defined by Assumption 7. If all nodes perform local computations with probability p_{comp}/n with $p_{\text{comp}} > p_{\text{comm}}^{\max}$ or if $\tau > 1$ then there exists $C < 24$ such that:*

$$\mathbb{P}\left(\frac{1}{t}T(t) \leq \frac{C}{n}\left(p_{\text{comp}} + 2\tau p_{\text{comm}}^{\max}\right)\right) \to 1 \text{ as } t \to \infty \tag{3.7.1}$$

Note that the constant C is a worst-case estimate and that it is much smaller for homogeneous communication probabilities. This novel result states that the number of iterations that the pairwise instanciation of Algorithm 4 can perform per unit of time increases linearly with the size of the network. This is possible because each iteration only involves two nodes so many iterations can be done in parallel. The assumption $p_{\text{comp}} > p_{\text{comm}}$ is responsible for the $1 + \tau$ factor instead of τ in Table 1, which prevents ADFS from benefiting from *network acceleration* when communications are cheap ($\tau < 1$). Note that this is an actual restriction of following a schedule, as detailed in Appendix 3.B. Yet, network operations generally suffer from communication protocols overhead whereas computing a single proximal update often either has a closed-form solution or is a simple one-dimensional problem in the linear case. Therefore, assuming $\tau > 1$ is not very restrictive in the finite-sum setting.

3.7.2. Performances and Parameters Choice in the Homogeneous Setting. We now prove the time to convergence of ADFS presented in Table 1, and detail the conditions under which it holds. Indeed, Section 3.6.2 presents ADFS in full generality but the different parameters have to be chosen carefully to reach optimal speed. In particular, we have to choose the coefficients μ to make sure that the graph augmentation trick does not cause the smallest positive eigenvalue of $A^T \Sigma^{-1} A$ to shrink too much. Similarly, ρ is defined by a minimum over all edges of a given quantity. This quantity heavily depends on whether the edge is an actual communication edge or a virtual edge. One can trade p_{comp} for p_{comm} so that the minimum is the same for both kind of edges, but Theorem 17 tells us that this is only possible as long as $p_{\text{comp}} > p_{\text{comm}}$.

Parameters choice. We define $L = A_{\text{comm}} A_{\text{comm}}^T \in \mathbb{R}^{n \times n}$ the Laplacian of the communication graph, with $A_{\text{comm}} \in \mathbb{R}^{n \times E}$ such that $A_{\text{comm}} e_{k\ell} = \mu_{k\ell}(e^{(k)} - e^{(\ell)})$ for all edge $(k, \ell) \in E^{\text{comm}}$, the set of communication edges. We define $\tilde{\gamma} = \min_{(k,\ell) \in E^{\text{comm}}} \lambda_{\min}^+(L) n^2 / (\mu_{k\ell}^2 R_{k\ell} E^2)$. As shown in Appendix 3.C.1, $\tilde{\gamma} \approx \gamma$ for regular graphs such as the complete graph or the grid, justifying the use of γ instead of $\tilde{\gamma}$ in Table 1. We assume for simplicity that $\sigma_i = \sigma$ and that $\kappa_i = 1 + \sigma_i^{-1} \sum_{j=1}^m L_{ij} = \kappa_s$ for all nodes. For virtual edges, we choose $\mu_{ij}^2 = \lambda_{\min}^+(L) L_{ij} / (\sigma \kappa_i)$ and $p_{ij} = p_{\text{comp}}(1 + L_{ij}\sigma_i^{-1})^{\frac{1}{2}} / (n S_{\text{comp}})$ with $S_{\text{comp}} = n^{-1} \sum_{i=1}^n \sum_{j=1}^m (1 + L_{ij}\sigma_i^{-1})^{\frac{1}{2}}$. This

corresponds to using a standard importance sampling scheme for selecting samples. For communications edges $(k,\ell) \in E^{\text{comm}}$, we choose uniform $p_{k\ell} = p_{\text{comm}}/E$ and $\mu_{k\ell}^2 = 1/2$.

Parameters tuning. The previous paragraph specifies relevant choices of parameters $\mu_{k\ell}$ and $p_{k\ell}$. Therefore, ADFS can be run *without manual tuning*. Extra tuning (such as communication probabilities) could be performed to adapt to specific heterogeneous situations. Yet, this should be considered as an extra degree of freedom that other algorithms may not have access to rather than an extra parameter to tune. For example, the choice of uniform communication probabilities is automatically enforced by synchronous gossip-based algorithms such as MSDA or DSBA (all edges are activated at each step). Note that choosing different values of $\mu_{k\ell}$ for communication edges amounts to tuning the gossip matrix, which is generally considered as an input of the problem.

THEOREM 18. *If we choose* $p_{\text{comm}} = \min\left(1/2, \left(1 + S_{\text{comp}}\sqrt{\tilde{\gamma}/\kappa_s}\right)^{-1}\right)$. *Then, running Algorithm 4 under the randomized sampling defined by Assumption 7 for* $K_\varepsilon = \rho^{-1}\log(\varepsilon^{-1})$ *iterations guarantees* $\mathbb{E}\left[\|\theta_{K_\varepsilon} - \theta^\star\|^2\right] \leq C_0\varepsilon$, *and takes time* $T(K_\varepsilon)$, *with:*

$$T(K_\varepsilon) \leq \sqrt{2}C\left(m + \sqrt{m\kappa_s} + \sqrt{2}\left(1 + 4\tau\right)\sqrt{\frac{\kappa_s}{\tilde{\gamma}}}\right)\log\left(1/\varepsilon\right)$$

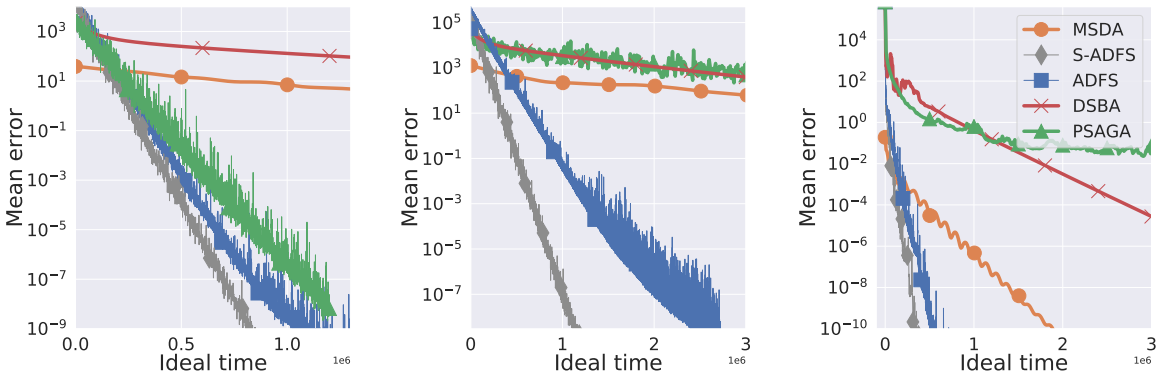
with probability tending to 1 *as* $\rho^{-1}\log(\varepsilon^{-1}) \to \infty$, *with* C_0 *and* C *as in Theorems 14 and 17.*

Theorem 18 assumes that all communication probabilities and condition numbers are exactly equal in order to ease reading. A more detailed version with rates for more heterogeneous settings can be found in Appendix 3.C. Note that while algorithms such as MSDA required to use polynomials of the initial gossip matrix to model several consecutive communication steps, we can more directly tune the amount of communication and computation steps simply by adjusting p_{comp} and p_{comm}. Note that the results in this Section are directly extracted from Hendrikx et al. [2019b] and thus obtain a dependence on κ_s instead of κ_{comm}. Yet, it is possible to refine the analysis as it was done in the synchronous case and obtain a dependence on κ_{comm}.

3.8. Experiments

In this section, we illustrate the theoretical results by showing how ADFS compares with MSDA [Scaman et al., 2017b], Point-SAGA [Defazio, 2016], and DSBA [Shen et al., 2018]. We also compare the synchronous version of ADFS (S-ADFS) to the locally synchronous one of the conference paper (ADFS) Hendrikx et al. [2019b]. All algorithms (except for DSBA, for which we fine-tuned the step-size) were run with out-of-the-box hyperparameters given by theory on data extracted from the standard Higgs and Covtype datasets from LibSVM[1]. To obtain the local dataset $X_i \in \mathbb{R}^{m\times d}$ of node i, m samples are drawn at random, so local datasets of different nodes may overlap. We used the logistic loss with quadratic regularization, meaning that the function at node i is $f_i : \frac{1}{m}\theta_i \mapsto \sum_{j=1}^m \log\left(1 + \exp(-l_{ij}X_{ij}^\top\theta_i)\right) + \frac{\sigma_i}{2}\|\theta_i\|^2$, where $l_{ij} \in \{-1, 1\}$ is the label associated with X_{ij},

[1]Datasets are available at https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/binary.html



(a) Higgs, $n = 4$ (b) Higgs, $n = 100$ (c) Covtype, $n = 100$

FIGURE 3. Performances of various decentralized algorithms on the logistic regression task with $m = 10^4$ points per node, regularization parameter $\sigma = 1$ and communication delays $\tau = 5$ on 2D grid networks of different sizes.

the k-th sample of node i. We then plot the average error $n^{-1} \sum_{i=1}^{n} \sum_{i'=1}^{n} f_{i'}(\theta_i)$. We chose $m = 10^4$ and $\sigma = 10^{-4}$ for all simulations. The underlying graph is a 2D grid network. Experiments were run in a distributed manner on an actual computing cluster. Yet, plots are shown for *idealized times* (computation delays are equal to 1 and communication delays are equal to $\tau = 5$) in order to abstract implementation details and ensure that reported timings were not impacted by the cluster status.

First of all, we note on all the plots from Figure 3 that ADFS and S-ADFS exhibit very similar performances. S-ADFS is always slightly faster because it suffers from no waiting time but, as argued in the conference paper Hendrikx et al. [2019b], the waiting time due to the local synchrony of ADFS is rather small. Although S-ADFS is easier to implement (series of synchronous rounds), it offers less flexibility than ADFS to deal with identified stragglers. In the next paragraph, we refer to both S-ADFS and ADFS as ADFS since the differences are rather small.

Figure 3(a) shows that, as predicted by theory, ADFS and Point-SAGA have similar rates on small networks. In this case, ADFS uses more computing power but has a small overhead. Figures 3(b) and 3(c) use a much larger grid to evaluate how these algorithms scale. In this setting, Point-SAGA is the slowest algorithm since it has 100 times less computing power available. MSDA performs quite well on the Covtype dataset thanks to its very good network scaling. Yet, the $m\sqrt{\kappa}$ factor in its rate makes it scales poorly with the condition number κ, which explains why it struggles on the Higgs dataset. DSBA is slow as well despite the fine-tuning because it is the only non-accelerated method, and it has to communicate after each proximal step, thus having to wait for a time $\tau = 5$ at each step. ADFS does not suffer from any of these drawbacks and outperforms other approaches by a large margin on these experiments. This illustrates the fact that ADFS combines the strengths of accelerated stochastic algorithms, such as Point-SAGA, and fast decentralized algorithms, such as MSDA.

3.9. Conclusion

In this paper, we develop an algorithmic framework for accelerated decentralized stochastic optimization based on accelerated block coordinate descent with arbitrary sampling. It is an extension of the conference paper [Hendrikx et al., 2019b] that provides stronger convergence results, and allows more flexibility in the algorithm design. This flexibility is obtained thanks to the arbitrary block sampling, so it is possible to transparently use global synchronous communications as well as local pairwise communications, or anything in between. The rate of ADFS explicitly mixes optimization-related and graph-related quantities, so that is is possible to adapt the parameters of the algorithm to heterogeneous problems with specific structure, as done with the line graph for instance.

We also provide a lower bound for decentralized stochastic optimization, and show that a synchronous implementation of ADFS almost matches this lower bound when parameters are chosen in a suitable way. The bound is exactly matched for generalized linear models, and otherwise a small gap due to the difference between the stochastic and batch condition numbers may exist. The problem of closing this gap in full generality remains open.

3.A. Accelerated Proximal Block Coordinate Descent with Arbitrary Sampling

3.A.1. Missing proofs. In this Section, we give detailed proofs of Theorem 13 and Corrolary 7. Before starting the actual proof, we define for $v \in \mathbb{R}$:

$$V_i^t(v) = \frac{B_{t+1}p_i}{2a_{t+1}}\|v - w_t^{(i)} + \eta_i e_i^\top \nabla f(y_t)\|^2 + \psi_i(v). \tag{3.A.1}$$

Then, we give the following lemma, which generalizes the proofs in Lin et al. [2015b] and Fercoq and Richtárik [2015] by considering non-uniform probabilities and that works with blocks of coordinates for both the convex and the strongly convex cases.

LEMMA 12. *If either* $1 - \beta_t - \frac{\alpha_t}{p_i} \geq 0$ *or* $\alpha_t = \beta_t$ *and* $1 - \frac{\alpha_t}{p_i} \geq 0$ *for any* i *such that* $\psi_i \neq 0$, *then for any* t *and* i *such that* $\psi_i \neq 0$, *we can write* $x_t^{(i)} = \sum_{l=0}^t \delta_t^{(i)}(l) v_l^{(i)}$ *such that* $\sum_{l=0}^t \delta_t^{(i)}(l) = 1$ *and for any* l, $\delta_t^{(i)}(l) \geq 0$. *We define* $\hat{\psi}_t^{(i)} = \sum_{l=0}^t \delta_t^{(i)}(l)\psi_i(v_l^{(i)})$ *and* $\hat{\psi}_t = \sum_{i=1}^d \hat{\psi}_t^{(i)}$. *Then, if* $R_i = 1$ *whenever* $\psi_i \neq 0$, $\psi(x_t) \leq \hat{\psi}_t$ *and:*

$$\mathbb{E}_{i_t}\left[\hat{\psi}_{t+1}\right] \leq \alpha_t \psi(\tilde{v}_{t+1}) + (1 - \alpha_t)\hat{\psi}_t. \tag{3.A.2}$$

where $\tilde{v}_{t+1}^{(i)} = \arg\min_v V_i^t(v)$ *for all* i. *In particular,* $v_{t+1}^{(i)} = \tilde{v}_{t+1}^{(i)}$ *if* $i \in b_t$ *and* $v_{t+1}^{(i)} = w_t^{(i)}$ *if* $i \notin b_t$.

PROOF OF LEMMA 12. This lemma is a generalization of a part of the APCG to arbitrary probabilities (instead of uniform ones). It still uses the fact that x_t can be written as a convex combination of $(v_l)_{l \leq t}$, but it requires to use a different convex combination for each coordinate of x_t, thus crucially exploiting separability of the proximal term. If coordinate i is such that $\psi_i = 0$, then $\hat{\psi}_{t+1}^{(i)} \leq \alpha_t \psi_i(\tilde{v}_{t+1}^{(i)}) + (1 - \alpha_t)\hat{\psi}_t^{(i)}$ is automatically satisfied for any $\delta_t^{(i)}$. For coordinates i such that $\psi_i \neq 0$ (and so $R_i = 1$), we start by expressing x_{t+1} in terms of x_t, v_{t+1} and v_t . More precisely, we write that for any $t > 0$:

$$x_{t+1}^{(i)} = y_t^{(i)} + \frac{\alpha_t}{p_i}(v_{t+1}^{(i)} - w_t^{(i)}).$$

Indeed, either coordinate i is updated at time t or $v_{t+1}^{(i)} = w_t^{(i)}$ so the previous equation always holds. We can then develop the w_t and y_t terms to obtain $x_{t+1}^{(i)}$ only in function of $x_t^{(i)}$, $v_t^{(i)}$ and $v_{t+1}^{(i)}$:

$$\begin{aligned}
x_{t+1}^{(i)} &= \frac{\alpha_t}{p_i} v_{t+1}^{(i)} + \left(1 - \frac{\alpha_t \beta_t}{p_i}\right) y_t^{(i)} - \frac{\alpha_t(1-\beta_t)}{p_i} v_t^{(i)} \\
&= \frac{\alpha_t}{p_i} v_{t+1}^{(i)} + \left(1 - \frac{\alpha_t \beta_t}{p_i}\right) \frac{(1-\alpha_t) x_t^{(i)} + \alpha_t(1-\beta_t) v_t^{(i)}}{1 - \alpha_t \beta_t} - \frac{\alpha_t(1-\beta_t)}{p_i} v_t^{(i)} \\
&= \frac{\alpha_t}{p_i} v_{t+1}^{(i)} + \alpha_t(1-\beta_t) \left[\frac{1 - \frac{\alpha_t \beta_t}{p_i}}{1 - \alpha_t \beta_t} - \frac{1}{p_i}\right] v_t^{(i)} + \left(1 - \frac{\alpha_t \beta_t}{p_i}\right) \frac{(1-\alpha_t)}{1 - \alpha_t \beta_t} x_t^{(i)} \\
&= \frac{\alpha_t}{p_i} v_{t+1}^{(i)} + \frac{\alpha_t(1-\beta_t)}{1 - \alpha_t \beta_t} \left(1 - \frac{1}{p_i}\right) v_t^{(i)} + \left(1 - \frac{\alpha_t \beta_t}{p_i}\right) \frac{(1-\alpha_t)}{1 - \alpha_t \beta_t} x_t^{(i)}.
\end{aligned}$$

At this point, all coefficients sum to 1. Indeed, they all sum to 1 at the first line and we have expressed $w_t^{(i)}$ and then $y_t^{(i)}$ as convex combinations of other terms, thus keeping the value of the sum unchanged. Yet, $p_i < 1$ so the coefficient on the second term is negative. Fortunately, it is possible to show that the $v_t^{(i)}$ term in the decomposition of $x_t^{(i)}$ is large enough so that the $v_t^{(i)}$ term in the decomposition of $x_{t+1}^{(i)}$ is positive. More precisely, we now show by recursion that for $t \geq 0$:

$$x_{t+1}^{(i)} = \frac{\alpha_t}{p_i} v_{t+1}^{(i)} + \sum_{l=0}^{t} \delta_{t+1}^{(i)}(l) v_l^{(i)}, \tag{3.A.3}$$

with $\delta_{t+1}^{(i)}(l) \geq 0$ for $l \leq t$. For $t = 0$, $x_0 = v_0$ and $x_1^{(i)} = \frac{\alpha_0}{p_i} v_1^{(i)} + \left(1 - \frac{\alpha_0}{p_i}\right) v_0^{(i)}$. We now assume that Equation (3.A.3) holds for a given $t > 0$, and expand $\delta_{t+1}^{(i)}(t)$ to show that it is positive. Using that $\delta_t^{(i)}(t) = \frac{\alpha_t}{p_i}$, we write:

$$\begin{aligned}
\delta_{t+1}^{(i)}(t) =& \frac{\alpha_t(1-\beta_t)}{1 - \alpha_t \beta_t} \left(1 - \frac{1}{p_i}\right) + \frac{\alpha_t}{p_i} \left(1 - \frac{\alpha_t \beta_t}{p_i}\right) \frac{(1-\alpha_t)}{1 - \alpha_t \beta_t} \\
&= \frac{\alpha_t}{1 - \alpha_t \beta_t} \left[(1-\beta_t)\left(1 - \frac{1}{p_i}\right) + \frac{(1-\alpha_t)}{p_i}\left(1 - \frac{\alpha_t \beta_t}{p_i}\right)\right] \\
&= \frac{\alpha_t}{1 - \alpha_t \beta_t} \left[1 - \beta_t - \frac{1}{p_i} + \frac{\beta_t}{p_i} + \frac{1}{p_i} - \frac{\alpha_t}{p_i} - (1-\alpha_t)\frac{\alpha_t \beta_t}{p_i^2}\right] \\
&= \frac{\alpha_t}{1 - \alpha_t \beta_t} \left[\left(1 - \beta_t - \frac{\alpha_t}{p_i}\right) + \frac{\beta_t}{p_i}\left(1 - (1-\alpha_t)\frac{\alpha_t}{p_i}\right)\right].
\end{aligned}$$

We conclude that $\delta_{t+1}^{(i)}(t) \geq 0$ since $1 - \beta_t - \frac{\alpha_t}{p_i} \geq 0$. Note that this condition can be weakened to $1 - \frac{\alpha_t^2}{p_i^2} \geq 0$ when $\beta_t = \alpha_t$ or when $\beta_t = 0$. We also deduce from the form of $x_{t+1}^{(i)}$ that for $l < t$, the only coefficients on $v_l^{(i)}$ in the development of $x_{t+1}^{(i)}$ come from the $x_t^{(i)}$ term and so:

$$\delta_{t+1}^{(i)}(l) = \left(1 - \frac{\alpha_t \beta_t}{p_i}\right) \frac{(1-\alpha_t)}{1 - \alpha_t \beta_t} \delta_t^{(i)}(l), \tag{3.A.4}$$

so these coefficients are positive as well. Since they also sum to 1, it implies that $x_t^{(i)}$ is a convex combination of the $v_l^{(i)}$ for $l \leq t$, and we use the convexity of ψ_i to write:

$$\psi_i(x_t^{(i)}) = \psi_i\left(\sum_{l=0}^{t} \delta_t^{(i)}(l) v_l^{(i)}\right) \leq \sum_{l=0}^{t} \delta_t^{(i)}(l)\psi_i(v_l^{(i)}) = \hat{\psi}_t^{(i)}.$$

Now, we can properly express $\hat{\psi}_{t+1}^{(i)}$ using the decomposition of $x_{t+1}^{(i)}$ in terms of $\delta_{t+1}^{(i)}$:

$$\begin{aligned}
\mathbb{E}[\hat{\psi}_{t+1}^{(i)}] &= \mathbb{E}[\frac{\alpha_t}{p_i}\psi_i(v_{t+1}^{(i)})] + \frac{\alpha_t(1-\beta_t)}{1-\alpha_t\beta_t}\left(1-\frac{1}{p_i}\right)\psi_i(v_t^{(i)}) \\
&+ \left(1-\frac{\alpha_t\beta_t}{p_i}\right)\frac{1-\alpha_t}{1-\alpha_t\beta_t}\sum_{l=0}^{t}\delta_t^{(i)}(l)\psi_i(v_l^{(i)}) \\
&= \alpha_t\psi_i(\tilde{v}_{t+1}^{(i)}) + (1-p_i)\frac{\alpha_t}{p_i}\psi_i(w_t^{(i)}) + \frac{\alpha_t(1-\beta_t)}{1-\alpha_t\beta_t}\left(1-\frac{1}{p_i}\right)\psi_i(v_t^{(i)}) \\
&+ \left(1-\frac{\alpha_t\beta_t}{p_i}\right)\frac{1-\alpha_t}{1-\alpha_t\beta_t}\hat{\psi}_t^{(i)}
\end{aligned}$$

At this point, we use the convexity of ψ_i to develop $\psi_i(w_t^{(i)})$ and then $\psi_i(y_t^{(i)})$ in the following way:

$$\begin{aligned}
\psi_i(w_t^{(i)}) &\leq (1-\beta_t)\psi_i(v_t^{(i)}) + \beta_t\psi_i(y_t^{(i)}) \\
&\leq (1-\beta_t)\psi_i(v_t^{(i)}) + \frac{\beta_t}{1-\alpha_t\beta_t}\left[(1-\alpha_t)\psi_i(x_t^{(i)}) + \alpha_t(1-\beta_t)\psi_i(v_t^{(i)})\right] \\
&= \frac{1-\beta_t}{1-\alpha_t\beta_t}\psi_i(v_t^{(i)}) + \frac{\beta_t(1-\alpha_t)}{1-\alpha_t\beta_t}\psi_i(x_t^{(i)}).
\end{aligned}$$

If we plug these expressions into the development of $\mathbb{E}[\hat{\psi}_{t+1}^{(i)}]$, the $\psi_i(v_t^{(i)})$ terms cancel and we obtain:

$$\mathbb{E}[\hat{\psi}_{t+1}^{(i)}] \leq \alpha_t\psi_i(\tilde{v}_{t+1}^{(i)}) + \alpha_t\left(\frac{1}{p_i}-1\right)\frac{\beta_t(1-\alpha_t)}{1-\alpha_t\beta_t}\psi_i(x_t^{(i)}) + \left(1-\frac{\alpha_t\beta_t}{p_i}\right)\frac{1-\alpha_t}{1-\alpha_t\beta_t}\hat{\psi}_t^{(i)}$$

We now use the fact that $\psi_i(x_t^{(i)}) \leq \hat{\psi}_t^{(i)}$ (by convexity of ψ_i) to get:

$$\begin{aligned}
\mathbb{E}[\hat{\psi}_{t+1}^{(i)}] &\leq \alpha_t\psi_i(\tilde{v}_{t+1}^{(i)}) + \frac{1-\alpha_t}{1-\alpha_t\beta_t}\left[\alpha_t\beta_t\left(\frac{1}{p_i}-1\right) + \left(1-\frac{\alpha_t\beta_t}{p_i}\right)\right]\hat{\psi}_t^{(i)} \\
&\leq \alpha_t\psi_i(\tilde{v}_{t+1}^{(i)}) + (1-\alpha_t)\hat{\psi}_t^{(i)}
\end{aligned}$$

This holds for any coordinate i and so $\mathbb{E}[\hat{\psi}_{t+1}] \leq \alpha_t\psi(\tilde{v}_{t+1} + (1-\alpha_t)\hat{\psi}_t$ for all $t \geq 0$, which finishes the proof of the lemma. □

We now introduce and prove Lemma 13, which is the main inequality from which the rest of the proof follows directly.

LEMMA 13. *For any block of coordinates b, the following inequality holds:*

$$\begin{aligned}\frac{1}{2\eta_t}[\|v_{t+1}-\theta^\star\|^2_{A^\dagger A}+\|v_{t+1}-w_t\|^2_{A^\dagger A}-\|\theta^\star-w_t\|^2_{A^\dagger A}]\\ \leq \langle P_b^\dagger \nabla q_A(y_t), \theta^\star - v_{t+1}\rangle_{A^\dagger A} + \sum_{i\in b}\frac{1}{p_i}\left[\psi_i\left(\theta^{\star(i)}\right)-\psi_i\left(v_{t+1}^{(i)}\right)\right].\end{aligned} \tag{3.A.5}$$

PROOF. We note $v_{t+1}^\perp$ the restriction of v_{t+1} to coordinates i such that $\psi_i = 0$. Similarly, we note $b^\perp$ the restriction of the block b to coordinates i such that $\psi_i = 0$. With these notations, we write:

$$\begin{aligned}\frac{1}{2\eta_t}[\|v_{t+1}^\perp-\theta^{\star\perp}\|^2_{A^\dagger A}+\|v_{t+1}^\perp-w_t^\perp\|^2_{A^\dagger A}-\|\theta^{\star\perp}-w_t^\perp\|^2_{A^\dagger A}]\\ \leq \sum_{i\in b^\perp}\frac{1}{p_i}\left[\langle\nabla_i q_A(y_t),\theta^\star-v_{t+1}\rangle_{A^\dagger A}+\psi_i\left(\theta^{\star(i)}\right)-\psi_i\left(v_{t+1}^{(i)}\right)\right].\end{aligned} \tag{3.A.6}$$

Equation (3.A.6) follows directly from using $v_{t+1}^\perp = w_t^\perp - \sum_{i\in b^\perp}\frac{\eta_t}{p_i}\nabla_i q_A(y_t)$ and basic algebra (expanding the squared terms).

If $i \in b$ is such that $\psi_i \neq 0$, we use the strong convexity of V_i^t at points $v_{t+1}^{(i)}$ (its minimizer, by definition) and $\theta^{\star(i)}$ (i-th coordinate of a minimizer of F) to write that $V_i^t(v_{t+1}^{(i)}) + \frac{B_{t+1}p_i}{2a_{t+1}}\|v_{t+1}^{(i)}-\theta^{\star(i)}\|^2 \leq V_i^t(\theta^{\star(i)})$. This is a key step from the proof of Lin et al. [2015b] and uses the same arguments as Lemma 3 from Fercoq and Richtárik [2015]. Then, expanding the V_i^t terms yields:

$$\begin{aligned}&\|v_{t+1}^{(i)}-\theta^{\star(i)}\|^2+\|v_{t+1}^{(i)}-w_t^{(i)}+\frac{a_{t+1}}{B_{t+1}p_i}\nabla_i q_A(y_t)\|^2\\ &-\|\theta^{\star(i)}-w_t^{(i)}+\frac{a_{t+1}}{B_{t+1}p_i}\nabla_i q_A(y_t)\|^2 \leq \frac{2a_{t+1}}{B_{t+1}p_i}\left[\psi_i\left(\theta^{\star(i)}\right)-\psi_i\left(v_{t+1}^{(i)}\right)\right].\end{aligned}$$

If we pull gradient terms out of the squares this yields:

$$\begin{aligned}\frac{1}{2\eta_t}[\|v_{t+1}^{(i)}-\theta^{\star(i)}\|^2_{A^\dagger A}+\|v_{t+1}^{(i)}-w_t^{(i)}\|^2_{A^\dagger A}-\|\theta^{\star(i)}-w_t^{(i)}\|^2_{A^\dagger A}]\\ \leq \frac{1}{p_i}\left[\langle\nabla_i q_A(y_t),\theta^\star-v_{t+1}\rangle_{A^\dagger A}+\psi_i\left(\theta^{\star(i)}\right)-\psi_i\left(v_{t+1}^{(i)}\right)\right].\end{aligned} \tag{3.A.7}$$

Finally, if $i \notin b$ is such that $\psi_i \neq 0$ then $v_{t+1}^{(i)} = w_t^{(i)}$ and so

$$\|v_{t+1}^{(i)}-\theta^{\star(i)}\|^2_{A^\dagger A}+\|v_{t+1}^{(i)}-w_t^{(i)}\|^2_{A^\dagger A}-\|\theta^{\star(i)}-w_t^{(i)}\|^2_{A^\dagger A}=0.$$

Note that $A^\dagger A e_i = e_i$ for all i such that $\psi_i \neq 0$ since $e_i^\top A^\dagger A e_i = 1$ and $A^\dagger A$ is a projector. Therefore,

$$\|v_{t+1}-\theta^\star\|^2_{A^\dagger A} = \|v_{t+1}^\perp-\theta^{\star\perp}\|^2_{A^\dagger A}+\sum_{i,\psi_i\neq 0}\|v_{t+1}^{(i)}-\theta^{\star(i)}\|^2_{A^\dagger A} \tag{3.A.8}$$

so we can sum Equation (3.A.6) with Equation (3.A.7) for all i to finish the proof. □

Now that we have detailed the key lemmas presented in the proof sketch, we can proceed to the actual proof.

Proof of Theorem 13. This proof follows the same general structure as Nesterov and Stich [Nesterov and Stich, 2017]. In particular, it follows from expanding the $\|v_{t+1} - \theta^\star\|^2$ term. In the original proof, $v_{t+1} = w_t - g$ where g is a gradient term so the expansion is rather straightforward. In our case, v_{t+1} is defined by a proximal mapping so a bit more work is required. Yet, similar terms appear, along with the function values of the non-smooth term that we control with Lemma 12. This expansion is done by Lemma 13, which relies on using the strong convexity of the proximal mapping.

We now evaluate each term of Equation (3.A.5). First of all, we use that $y_t - x_{t+1} = \alpha_t P_b^\dagger A^\dagger A(w_t - v_{t+1})$ to write:

$$\begin{aligned}
&\mathbb{E}[\langle P_b^\dagger \nabla q_A(y_t), \theta^\star - v_{t+1}\rangle_{A^\dagger A}] \\
&= \mathbb{E}[\langle P_b^\dagger \nabla q_A(y_t), \theta^\star - w_t\rangle_{A^\dagger A}] + \mathbb{E}[\nabla q_A(y_t)^\top P_b^\dagger A^\dagger A(w_t - v_{t+1})] \\
&= \langle \nabla q_A(y_t), \theta^\star - w_t\rangle_{A^\dagger A} + \alpha_t^{-1}\mathbb{E}[\nabla q_A(y_t)^\top (y_t - x_{t+1})].
\end{aligned}$$

The rest of this proof closely follows the analysis from Hendrikx et al. [Hendrikx et al., 2019a], which is an adaptation of Nesterov and Stich [Nesterov and Stich, 2017] to strong convexity on a subspace. The main difference is that it is also necessary to control the function values of ψ, which is done using Lemma 12. For the first term, we use the strong convexity of f as well as the fact that $w_t = y_t - \frac{1-\alpha_t}{\alpha_t}(x_t - y_t)$ to obtain:

$$\begin{aligned}
a_{t+1}\nabla q_A(y_t)^\top A^\dagger A(\theta^\star - w_t) &= a_{t+1}\nabla q_A(y_t)^\top A^\dagger A\left(\theta^\star - y_t + \frac{1-\alpha_t}{\alpha_t}(x_t - y_t)\right) \\
&\leq a_{t+1}\left(q_A(\theta^\star) - q_A(y_t) - \frac{1}{2}\sigma_A\|y_t - \theta^\star\|^2_{A^\dagger A} + \frac{1-\alpha_t}{\alpha_t}(q_A(x_t) - q_A(y_t))\right) \\
&\leq a_{t+1}q_A(\theta^\star) - A_{t+1}q_A(y_t) + A_t q_A(x_t) - \frac{1}{2}a_{t+1}\sigma_A\|y_t - \theta^\star\|^2_{A^\dagger A}.
\end{aligned}$$

For the second term we use the smoothness of q_A and then the fact that $x_{t+1} - y_t$ has support on U_k only (just like $v_{t+1} - w_t$), as well as the fact that $A^\dagger A$ is symmetric to obtain:

$$\begin{aligned}
&\frac{a_{t+1}}{\alpha_t}\nabla q_A(y_t)^\top (y_t - x_{t+1}) \\
&\leq A_{t+1}\left[q_A(y_t) - q_A(x_{t+1})\right] + \frac{a_{t+1}}{2\alpha_t}\|x_{t+1} - y_t\|^2_M \\
&\leq A_{t+1}\left[q_A(y_t) - q_A(x_{t+1})\right] + \frac{a_{t+1}\alpha_t}{2}\|A^\dagger A(v_{t+1} - w_t)\|^2_{P_b^\dagger M P_b^\dagger} \\
&\leq A_{t+1}\left[q_A(y_t) - q_A(x_{t+1})\right] + \frac{a_{t+1}^2\lambda_{\max}((A^\dagger A P_b^\dagger M P_b^\dagger A^\dagger A)}{2A_{t+1}}\|v_{t+1} - w_t\|^2_{A^\dagger A}.
\end{aligned}$$

Noting $\Delta q_A(x_t) = \mathbb{E}[q_A(x_t)] - q_A(\theta^\star)$ and remarking that $a_{t+1} = A_{t+1} - A_t$, we obtain, using that $\alpha_t = \frac{a_{t+1}}{A_{t+1}}$:

$$\begin{aligned}
&\mathbb{E}[a_{t+1}\langle P_b^\dagger \nabla q_A(y_t), \theta^\star - v_{t+1}\rangle_{A^\dagger A}] \\
&\leq A_t\Delta q_A(x_t) - A_{t+1}\Delta q_A(x_{t+1}) + \frac{B_{t+1}}{2}\mathbb{E}[\|w_t - v_{t+1}\|^2_{A^\dagger A}] - \frac{a_{t+1}\sigma_A}{2}\|y_t - \theta^\star\|^2_{A^\dagger A}.
\end{aligned}$$

Using Lemma 12, we derive in the same way:

$$\mathbb{E}[\frac{a_{t+1}}{p_i}\left[\psi_i\left(\theta^{\star(i)}\right) - \psi_i\left(v_{t+1}^{(i)}\right)\right]]$$

$$= a_{t+1}\psi(\theta^\star) - A_{t+1}\alpha_t\psi(\tilde{v}_{t+1})$$
$$\leq A_t\left(\mathbb{E}[\hat{\psi}_t] - \psi(\theta^\star)\right) - A_{t+1}\left(\mathbb{E}[\hat{\psi}_{t+1}] - \psi(\theta^\star)\right).$$

Now, we can multiply Equation (3.A.5) by $\frac{a_{t+1}}{p_i}$ and take the expectation over i. The $\|v_{t+1} - w_t\|^2_{A^\dagger A}$ terms cancel and we obtain:

$$\frac{B_{t+1}}{2}\mathbb{E}[\|v_{t+1} - \theta^\star\|^2_{A^\dagger A}] + A_{t+1}\Delta\hat{F}_A(x_{t+1})$$
$$\leq A_t\Delta\hat{F}_A(x_t) + \frac{B_{t+1}}{2}\|w_t - \theta^\star\|^2_{A^\dagger A} - \frac{a_{t+1}\sigma_A}{2}\|y_t - \theta^\star\|^2_{A^\dagger A},$$

where $\Delta\hat{F}_A(x_t) = \Delta q_A(x_t) + \mathbb{E}[\hat{\psi}_t] - \psi(\theta^\star)$. Convexity of the squared norm yields $\|w_t - \theta^\star\|^2_{A^\dagger A} \leq (1-\beta_t)\|v_t - \theta^\star\|^2_{A^\dagger A} + \beta_t\|y_t - \theta^\star\|^2_{A^\dagger A}$. Now remarking that $B_{t+1}(1-\beta_t) = B_t$ and $a_{t+1}\sigma_A = B_{t+1}\beta_t$, and summing the inequalities until $t = 0$, we obtain:

$$B_t\|v_t - \theta^\star\|^2_{A^\dagger A} + 2A_t\Delta\hat{F}_A(x_t) \leq 2A_0\Delta F_A(x_0) + B_0\|v_0 - \theta^\star\|^2_{A^\dagger A}.$$

We finish the proof by using the fact that $\psi(x_t) \leq \hat{\psi}_t$ and $\psi(x_0) = \hat{\psi}_0$ since $x_0 = v_0$. □

Proof of Corollary 7. We first prove that α_t can actually be obtained by a simple recursion. This comes from the (well-known) fact that the recursions in Lin et al. [2015b] and Nesterov and Stich [2017] are actually the same. If $\sigma_A = 0$ then we have to choose $\beta_t = 0$ for all t. Then, we can choose $B_t = B_0$ for any $B_0 > 0$. This allows to write $(A_{t+1} - A_t)^2 S^2 = A_t B_0$ for all t, which is a second degree polynomial in the variable A_{t+1}. We choose the positive root in order to have $a_{t+1} \geq 0$, which yields:

$$A_{t+1} = A_t + \frac{B_0}{2S^2}\left(1 + \sqrt{1 + 4S^2 B_0^{-1} A_t}\right). \tag{3.A.9}$$

Coefficients (a_t) can be computed using

$$a_{t+1} = A_{t+1} - A_t = \frac{B_0}{2S^2}\left(1 + \sqrt{1 + 4S^2 B_0^{-1} A_t}\right),$$

and so we use the fact that $a_{t+1}S^2 = A_{t+1}B_{t+1}$, which can be rewritten as $\alpha_t = \frac{B_0}{a_{t+1}S^2}$. to obtain the sequence (α_t) as:

$$\alpha_t = \frac{2}{1 + \sqrt{1 + 4S^2 B_0^{-1} A_t}}.$$

In particular,

$$A_t = \left[\left(\frac{2}{\alpha_t} - 1\right)^2 - 1\right]\frac{B_0}{4S^2}.$$

This expression for A_t and A_{t+1} can be substituted in the relation $A_{t+1} = A_t + \frac{B_0}{a_{t+1}S^2}$, which yields after some simplifications:

$$\alpha_{t+1}^{-2} - \alpha_{t+1}^{-1} - \alpha_t^{-2} = 0,$$

which is a second degree polynomial in the variable α_{t+1}^{-1}. Solving for α_t leads to

$$\alpha_{t+1} = \frac{2}{1 + \sqrt{1 + 4\alpha_t^{-2}}} = \frac{\sqrt{\alpha_t^4 + 4\alpha_t^2} - \alpha_t^2}{2},$$

which is the exact same recursion as in Lin et al. [2015b] and Fercoq and Richtárik [2015]. In particular, only the value of α_0 matters and only the sequence α_t actually needs to be computed, since the only coefficients needed are the α_t and $\frac{a_{t+1}}{B_{t+1}} = \frac{1}{\alpha_t S^2}$.

We would like to choose the highest possible α_0, such that $1 - \alpha_0/p_{\min} \geq 0$, so we take $\alpha_0 = p_{\min}$ where $p_{\min} = \min_i p_i$ where the minimum is over all coordinates such that $\psi_i \neq 0$. This is enough to respect the condition $\alpha_t \leq p_{\min}$ since (α_t) is a decreasing sequence. This leads to

$$A_0 = \left[\left(\frac{2}{p_{\min}} - 1\right)^2 - 1\right] \frac{B_0}{4S^2} \leq \frac{B_0}{p_{\min}^2 S^2}.$$

Since $A_0 \geq 0$, a direct recursion yields $A_t \geq \frac{B_0 t^2}{4S^2}$. We call $r_t^2 = \|v_0 - \theta_A^\star\|_{A^\dagger A}^2 - \mathbb{E}[\|v_t - \theta_A^\star\|_{A^\dagger A}^2]$, and $\Delta F_t = \mathbb{E}[q_A(x_t) + \psi(x_t)] - q_A(\theta_A^\star) + \psi_A(\theta_A^\star)$, then:

$$F_t \leq \frac{1}{2A_t}\left(B_0 r_t^2 + 2A_0 F_0\right) = \frac{B_0}{2A_t}\left(r_t^2 + \frac{2}{S^2 p_{\min}^2} F_0\right) \leq \frac{2S^2}{t^2}\left(r_t^2 + \frac{2}{S^2 p_{\min}^2} F_0\right),$$

which finishes the proof of the rate.

□

Sampling with replacement. The arbitrary sampling litterature for accelerated coordinate descent methods is vast [Lee and Sidford, 2013, Allen-Zhu et al., 2016, Nesterov and Stich, 2017, Hanzely and Richtárik, 2019], and we present in this paper results for the general setting of block proximal coordinate gradient. Yet, standard mini-batch stochastic gradient descent algorithms use sampling *with* replacement, whereas coordinate descent methods always use the notion of blocks, *i.e.*, *without* replacement. Algorithm 3 does not extend to sampling with replacement, and this mainly comes from the fact that proximal updates do not mix well with sampling with replacement, and Lemma 12 does not hold anymore in this case.

3.A.2. Efficient implementation. Our extended APCG algorithm is also closely related with an arbitrary sampling version of APPROX Fercoq and Richtárik [2015]. Similarly to Lee and Sidford Lee and Sidford [2013], APPROX also uses iterations that can be more efficient, especially in the linear case. These extensions can also be applied to APCG under the same assumptions, as shown in Lin et al. [2015b]. We present in this section the efficient implementations of the generalized APCG algorithm. The main goal is to avoid as much as possible to perform convex combinations of dense vectors. The main changes in Algorithm 6 are that we express the proximal operator of line 5 in a slightly different but equivalent form and that line 6 requires a matrix product to take into account the block aspect and the strong convexity in an arbitrary norm. The proof is a straightforward adaptation of Lin et al. [2015b]. Note that the full vector w_t never actually needs to be formed so local updates are sparse.

We now present the convex case, which is an adaptation of Fercoq and Richtárik [2015]. We therefore refer the interested reader to this paper for the details of the equivalence between Algorithm 3 in the convex case and Algorithm 7.

Algorithm 6 Efficient Generalized APCG(ρ, σ_A, p_i), Strongly Convex Case.

$u_0 = 0$, $z_0 = 0$, $\phi = \frac{1-\rho}{1+\rho}$, $\eta = \frac{\rho}{\sigma_A}$
while $t < T$ **do**
  $w_t = -\phi^{t+1} u_t + v_t$
  $g_t = \eta P_b^\dagger \nabla q_A(\phi^{t+1} u_t + z_t)$
  $h_t^{(i)} = \text{prox}_{\eta p_i^{-1} \psi_i}\left(w_t^{(i)} - g_t^{(i)}\right) - w_t^{(i)}$ for all $i \in b$, 0 otherwise.
  $u_{t+1} = u_t - \frac{I - \rho P_{b_t}^\dagger A^\dagger A}{2\phi^{t+1}} h_t, \quad z_{t+1} = z_t + \frac{I + \rho P_{b_t}^\dagger A^\dagger A}{2} h_t$
return $\phi^T u_T + z_T$

Algorithm 7 Efficient Generalized APCG(ρ, σ_A, p_i), Convex Case.

$u_0 = 0$, $v_0 = 0$, $\alpha_0 = \min_{i, \psi_i \neq 0} p_i$, $\eta_t = \frac{1}{\alpha_t S^2}$
while $t < T$ **do**
  $g_t = \eta_t P_b^\dagger \nabla q_A(\alpha_t^2 u_t + z_t)$
  $z_{t+1}^{(i)} = \text{prox}_{\eta_t p_i^{-1} \psi_i}\left(z_t^{(i)} - g_t^{(i)}\right)$ for all $i \in b$, 0 otherwise.
  $u_{t+1} = u_t - \frac{I - \alpha_t P_{b_t}^\dagger A^\dagger A}{2\alpha_t^2}(z_{t+1} - z_t)$
  $\alpha_{t+1} = \frac{\sqrt{\alpha_t^4 + 4\alpha_t^2} - \alpha_t^2}{2} = \frac{2}{1 + \sqrt{1 + 4\alpha_t^{-2}}}$
return $\alpha_{T-1}^2 u_T + v_T$

3.B. Average Time per Iteration

3.B.1. More communications implies more waiting. A fundamental assumption for Theorem 17 is to assume that $p_{\text{comm}} < p_{\text{comp}}$. In particular, it prevents p_{comm} from being too high since $p_{\text{comm}} + p_{\text{comp}} = 1$. Although this assumption seems quite restrictive in the first place, it is very intuitive to want to avoid p_{comm} from being too high, especially in the limit of $p_{\text{comm}} \to 1$ and τ arbitrarily small. Consider that one node (say node 0) starts a local update at some point. Communications are very fast compared to computations so it is very likely that the neighbors of node 0 will only perform communication updates, and they will do so until they have to perform one with node 0. At this point, they will have to wait until node 0 finishes its local computation, which can take a long time. Now that the neighbors of node 0 are also blocked waiting for the computation to finish, their neighbors will start establishing a dependence on them rather quickly. If the probability of computing is small enough and if the computing time is large enough, all nodes will sooner or later need to wait for node 0 to finish its local update before they can continue with the execution of their part of the schedule. In the end, only node 0 will actually be performing computations while all the others will be waiting.

This phenomenon is not restricted to the limit case presented above and the synchronization cost blows up as soon as $p_{\text{comm}} > p_{\text{comp}}$ and $\tau < 1$. In the proof below, the goal is to bound the total expected weight $\sum_{i=1}^n \mathbb{E}\left[X^t(i, w)\right]$ for w higher than a given threshold. Local computing operations will move mass from small values of w to higher values of w. On the other hand, communication operations will introduce synchronization between two nodes, thus increasing the total available mass $\sum_{w \geq 0} \sum_{i=1}^n \mathbb{E}\left[X^t(i, w)\right]$ (and not just moving

it to higher values of w) because it will duplicate the mass for $X^t(i,w)$ to $X^t(j,w)$ if nodes i and j communicate. This is the technical reason why $p_{\text{comm}} < p_{\text{comp}}$ is needed for this proof.

3.B.2. Detailed average time per iteration proof. The goal of this section is to prove Theorem 17. The proof is an extension of the proof of Theorem 2 from Hendrikx et al. [2019a]. Similarly, we denote t the number of iterations that the algorithm performs and τ_c^{ij} the random variable denoting the time taken by a communication on edge (i,j). Similarly, τ_l^i denotes the time taken by a local computation at node i. Then, 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 $w \in \mathbb{N}^*$:

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

where $\tau_c^{ij}(t)$ is the realization of τ_c^{ij} corresponding to the time taken by activating edge (i,j) at time t. If node i is chosen for a local computation, which happens with probability p_i^{comp} then $X^{t+1}(i, w + \tau_l^i(t)) = X^t(i,w)$ for all w. Otherwise, $X^{t+1}(j,w) = X^t(j,w)$ for all w. At time $t=0$, $X^0(i,0) = 1$ and $X^0(i,w) = 0$ for all w. Lemma 14 gives a bound on the probability that the time taken by the algorithm to complete t iterations is greater than a given value, depending on variables X^t. Note that a Lemma similar to the one by Hendrikx et al. [2019a] holds although variable X has been modified.

LEMMA 14. *We denote $T_{\max}(t)$ the time at which the last node of the system finishes iteration t. Then for all $\nu > 0$:*

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

PROOF. We first prove by induction on t that for any $i \in \{1,..,n\}$:

$$T_i(t) = \max_{w \in \mathbb{N}, X^t(i,w) > 0} w. \tag{3.B.1}$$

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 $w \in \mathbb{N}^*$, $X^{t+1}(i,w) = X^t(i,w)$ so the property is true. Besides, if $j \neq l$,

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

Similarly if $k = l$ (a local computation is performed at iteration t), then $w_{\max}(k,t+1) = \tau_l^k(t) + w_{\max}(k,t) = T_k(t) + \tau_l^k(t) = T_k(t+1)$. Then, we use the union bound and the the fact that having $X^t(i,w) > 0$ is equivalent to having $X^t(i,w) \geq 1$ since $X^t(i,w)$ is integer

valued to show that:

$$\mathbb{P}\left(T_{\max}(t) \geq \nu t\right) = \mathbb{P}\left(\max_{w, \sum_{i=1}^n X_i^t(w)>0} w \geq \nu t\right) \leq \mathbb{P}\left(\cup_{w \geq \nu t} \sum_{i=1}^n X_i^t(w) \geq 1\right) \leq \sum_{w \geq \nu t} \mathbb{P}\left(\sum_{i=1}^n X_i^t(w) \geq 1\right),$$

so using Markov inequality yields:

$$\mathbb{P}\left(T_{\max}(t) \geq \nu t\right) \leq \sum_{w \geq \nu t} \sum_{i=1}^n \mathbb{E}\left[X_i^t(w)\right]. \tag{3.B.2}$$

□

Variables X_i^t are obtained by linear recursions, so Lemma 14 allows us to bound the growth of variables with a simple recursion formula instead of evaluating a maximum. We write p_i^{comp} and p_i^{comm} the probability that node i performs a computation (respectively communication) update at a given time step, and $p_i = p_i^{\text{comp}} + p_i^{\text{comm}}$. We introduce $\underline{p}_{\text{comp}} = \min_i p_i^{\text{comp}}$ and $\bar{p}_{\text{comp}} = \max_i p_i^{\text{comp}}$ (and the same for communication probabilities).

LEMMA 15. *For all i, and all $\nu > 0$, if $\frac{1}{2} \geq \underline{p}_{\text{comp}} = \bar{p}_{\text{comp}} \geq \bar{p}_{\text{comm}}$ then:*

$$\sum_{w \geq (\nu_c + \nu_l)t} \sum_{i=1}^n \mathbb{E}\left[X^t(i,w)\right] \to 0 \text{ when } t \to \infty, \tag{3.B.3}$$

with $\nu_c = 6p_c\tau_c$ and $\nu_l = 9p_l\tau_l$ where $p_c = 4\bar{p}_{\text{comm}}$ and $p_l = \bar{p}_{\text{comp}}$.

Note that the constants in front of the ν parameters are very loose.

PROOF. Taking the expectation over the edges that can be activated gives, with $\tau_c^{ij}(\tau)$ the probability that τ_c^{ij} takes value τ (and the same for τ_l):

$$\begin{aligned}\mathbb{E}\left[X^{t+1}(i,w)\right] = &(1-p_i)\,\mathbb{E}\left[X^t(i,w)\right] \\ &+ p_{\text{comm}} \sum_{j=1}^n p_{ij} \sum_{\tau=0}^\infty \tau_c^{ij}(\tau)\left(\mathbb{E}\left[X^t(i,w-\tau)\right] + \mathbb{E}\left[X^t(j,w-\tau)\right]\right) \\ &+ p_i^{\text{comp}} \sum_{\tau=0}^\infty \tau_l^{ij}(\tau)\mathbb{E}\left[X^t(i,w-\tau)\right].\end{aligned}$$

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:

$$\bar{X}^{t+1}(w) = (1-\underline{p})\,\bar{X}^t(w) + 2\bar{p}_{\text{comm}} \sum_{\tau=0}^\infty \tau_c^{\max}(\tau)\bar{X}^t(w-\tau) + \bar{p}_{\text{comp}} \sum_{\tau=0}^\infty \tau_l^{\max}(\tau)\bar{X}^t(w-\tau).$$

with $\tau_c^{\max}(\tau) = \max_{ij} \tau_c^{ij}(\tau)$ (and the same for τ_l). We now introduce $\phi^t(z) = \sum_{w \in \mathbb{N}} z^w \bar{X}^t(w)$. We denote ϕ_c and ϕ_l the generating functions of $\tau_c^{\max}(\tau)$ and $\tau_l^{\max}(\tau)$. A direct recursion leads to:

$$\phi^t(z) = \left(1 - \underline{p}_{\text{comm}} - \underline{p}_{\text{comp}} + \bar{p}_{\text{comp}}\phi_l(z) + 2\bar{p}_{\text{comm}}\phi_c(z)\right)^t = \left(\phi^1(z)\right)^t.$$

We denote $\phi_{bin}(p,t)$ the generating function associated with the binomial law of parameters p and t. With this definition, we have:

$$\begin{aligned}&\phi_{bin}(p_c,t)(\phi_c(z))\phi_{bin}(p_l,t)(\phi_l(z)) = \\ &\qquad \left[(1-p_c)(1-p_l) + (1-p_c)p_l\phi_l(z) + (1-p_l)p_c\phi_c(z) + p_cp_l\phi_c(z)\phi_l(z)\right]^t,\end{aligned}$$

so we can define:

$$\phi_+^t(z) = (1+\delta)^t \phi_{bin}(p_c, t)(\phi_c(z))\phi_{bin}(p_l, t)(\phi_l(z)),$$

where p_c, p_l and δ are such that:

$$\frac{p_c}{1-p_c} \geq 2\frac{\bar{p}_{\text{comm}}}{1-\underline{\mathrm{p}}}, \quad \frac{p_l}{1-p_l} = \frac{\bar{p}_{\text{comp}}}{1-\underline{\mathrm{p}}} \text{ and } \delta \geq \frac{1-\underline{\mathrm{p}}}{(1-p_c)(1-p_l)} - 1.$$

Since $\bar{p}_{\text{comp}} = \underline{\mathrm{p}}_{\text{comp}}$ then $\underline{\mathrm{p}} \geq \bar{p}_{\text{comp}}$. Therefore, these conditions are satisfied for p_c and p_l as given by Lemma 15 and $\delta = (1-p_c)^{-1} - 1$. Then $(1+\delta)(1-p_c)(1-p_l) \geq 1-\underline{\mathrm{p}}$, $(1+\delta)(1-p_c)p_l \geq \bar{p}_{\text{comp}}$ and $(1+\delta)(1-p_l)p_c \geq 2\bar{p}_{\text{comm}}$. This means that if we write $\phi^1(z) = a_0 + a_c\phi_c(z) + a_l\phi_l(z)$ and $\phi_+^1(z) = b_0 + b_c\phi_c(z) + b_l\phi_l(z)$ then $b_0 \geq a_0$, $b_c \geq a_c$ and $b_l \geq a_l$. In particular, all the coefficients of ϕ^t are smaller than the coefficients of ϕ_+^t where both functions are integral series. Therefore, if we call Z_t the random variables associated with the generating function $(1+\delta)^{-t}\phi_+^t$ then for all i, t, w:

$$\mathbb{E}\left[X^t(i,w)\right] \leq (1+\delta)^t \mathbb{P}\left(Z_t = w\right), \tag{3.B.4}$$

where $Z_t = Z_c^t + Z_l^t = Bin(p_c, t)(Z_c) + Bin(p_l, Z_l)(\tau_l)$ where Z_c and Z_l are the random variables modeling the time of one communication or computation update. We can then use the bound $p(Z_t \geq (\nu_c + \nu_l)t) \leq p(Z_c^t \geq \nu_c t) + p(Z_l^t \geq \nu_l t)$. This way, we can bound the *communication* and *computation* costs independently. Then, we write a Chernoff bound, i.e. for any $\lambda > 0$:

$$\mathbb{P}\left(Z_c^t \geq \nu t\right) \leq e^{-\lambda\nu t}\mathbb{E}[e^{\lambda Z_c^t}] = e^{-\lambda\nu t}\mathbb{E}[e^{\lambda Z_c}]^t = e^{-\lambda\nu t}\left[1 - p_c + p_c\sum_{\tau=0}^{\infty} p_c(\tau)e^{\lambda\tau}\right]^t,$$

where S_c is the sum of t i.i.d. random variables drawn from τ_c. If $Z_c = \tau_c$ with probability 1 (deterministic delays) then this reduces to:

$$\mathbb{P}\left(Z_c^t \geq \nu_c t\right) \leq e^{-\lambda\nu_c t}\left[1 - p_c + p_c e^{\lambda\tau_c}\right].$$

Finally, we take $\nu_c = kp_c\tau_c$, $\lambda = \frac{1}{\tau_c}\ln(k)$ and we use the basic inequality $\ln(1+x) \geq \frac{x}{1+x}$ to show that:

$$-\ln\left[\mathbb{P}\left(Z_c^t \geq \nu_c t\right)\right] \geq t\left[\lambda\nu_c - p_c\left(e^{\lambda\tau_c} - 1\right)\right] \geq t(k(\ln(k) - 1) - 1)p_c.$$

Using the same log inequality and the fact that $p_c \geq \frac{1}{2}$ yields:

$$\ln(1+\delta) = -\ln(1-p_c) \leq \frac{p_c}{1-p_c} \leq 2p_c.$$

Therefore, choosing $k = 6$ ensures that $k(\ln(k) - 1) - 1 \geq 3$ and so:

$$(1+\delta)^t\mathbb{P}\left(Z_c^t \geq \nu_c t\right) \leq e^{-tp_c}.$$

We can apply the same reasoning to Z_l^t, and the bound is still valid with $k = 9$ because $p_l = \bar{p}_{\text{comp}} \geq \bar{p}_{\text{comm}} = p_c/4$. We finish the proof by using Equation (3.B.4). □

3.C. Algorithm Performances

ADFS has a linear convergence rate because it results from using generalized APCG. Yet, it is not straightforward to derive hyperparameters that lead to a rate that is fast and that can be easily interpreted. The goal of this section is to choose such parameters when the functions f_{ij} are smooth.

3.C.1. Eigengap $\tilde{\gamma}$. This section aims as justifying the $\tilde{\gamma}$ notation. Recall that it is defined such that $\tilde{\gamma} = \min_{(k,\ell)\in E^{\text{comm}}} \frac{\lambda^+_{\min}(L)n^2}{\mu^2_{k\ell} e^T_{k\ell} A^\dagger A e_{k\ell} E^2}$. We show in this section that for any given family of regular graphs, there exists a constant C_γ independent of the size of the graph such that $C_\gamma \tilde{\gamma} \geq \gamma$. Matrix A depends on μ, and we consider in this section that $\mu^2_{k\ell} = \mu_0$ for all communication edges (k,ℓ). Similar results can be obtained when μ is heterogeneous.

Regular graphs.. We say that a family of graph is regular if there exists $C_\gamma > 0$ such that $e^T_{k\ell} A^\dagger A e_{k\ell} \leq C_\gamma \frac{n}{E}$ for any $n > 2$.

Recall that E is the number of edges (usually constrained by the graph family and the number of nodes), and $e^T_{k\ell} A^\dagger A e_{k\ell}$ is the effective resistance of edge (k,ℓ). This assumption seems a bit technical but it simply requires that all edges contribute equally to the connectivity of the graph, and therefore is related to how symmetric the graph is. In particular, it is verified with $C_\gamma = 1$ for any completely symmetric graph, such as the complete graph or the ring. Since $e^T_{k\ell} A^\dagger A e_{k\ell} \leq 1$, it is also satisfied any time the ratio n/E is bounded below, and in particular for the grid, the hypercube, or any graph with bounded degree. Under these assumptions, and for any communication edge (k,ℓ):

$$\frac{\lambda^+_{\min}(L)n^2}{\mu^2_{k\ell} e^T_{k\ell} A^\dagger A e_{k\ell} E^2} \geq \frac{\gamma}{C_\gamma} \frac{\lambda_{\max}(L)n}{\mu^2_{k\ell} E} \geq \frac{\gamma}{C_\gamma} \frac{\text{Trace}(L)}{\mu_0^2 E} = 2\frac{\gamma}{C_\gamma}.$$

Here, we used the fact that $\text{Trace}(L) = 2\mu_0^2 E$, which can be deduced directly from the form of A (each edge has weight μ_0^2 and contributes two times, one for each end). We conclude by using the fact that since the previous inequalities are true for any $(k,\ell) \in E^{\text{comm}}$, it is in particular true for $\tilde{\gamma}$.

3.C.2. Communication rate and local rate. We know that the rate of ADFS can be written as the minimum of a given quantity over all edges of the graph. This quantity will be very different whether we consider communication edges or virtual edges. In this section, we give lower bounds for each type of edge, and show that we can trade one for the other by adjusting the probability of communication.

LEMMA 16. *With the choice of parameters of Theorem 18, parameter ρ satisfies:*

$$\rho \geq \frac{1}{\sqrt{2}n} \min\left(p_{\text{comm}} \Delta_p \sqrt{\frac{\tilde{\gamma}}{2\kappa}}, p_{\text{comp}} \frac{\sqrt{r_\kappa}}{S_{\text{comp}}} \right). \tag{3.C.1}$$

PROOF. Recall that the rate ρ is defined as:

$$\rho^2 = \min_{k\ell} \frac{p^2_{k\ell}}{\mu^2_{k\ell} e^T_{k\ell} A^\dagger A e_{k\ell}} \frac{\lambda^+_{\min}(\tilde{L})}{\sigma_k^{-1} + \sigma_\ell^{-1}}, \tag{3.C.2}$$

and that Lemma 9 ensures that

$$\lambda^+_{\min}(\tilde{L}) \geq \frac{\lambda^+_{\min}(L)}{2\sigma\kappa}.$$

Therefore, for communication edges the rate writes:

$$\rho_{\text{comm}}^2 \geq \min_{(k,\ell)\in E^{\text{comm}}} \left(\frac{1}{\sigma_k}+\frac{1}{\sigma_\ell}\right)^{-1} \frac{p_{k\ell}^2}{\mu_{k\ell}^2 e_{k\ell}^T A^\dagger A e_{k\ell}} \frac{\lambda_{\min}^+(L)}{2\sigma\kappa}. \tag{3.C.3}$$

If we take $\sigma_k = \sigma$ for all k and $p_{k\ell}^2 \geq \Delta_p^2 p_{\text{comm}}^2/|E|^2$ (corresponding to a homogeneous case), then we can make $\tilde{\gamma}$ appear to obtain:

$$\rho_{\text{comm}}^2 \geq \Delta_p^2 \frac{\tilde{\gamma}}{\kappa} \frac{p_{\text{comm}}^2}{4n^2}. \tag{3.C.4}$$

For "computation edges", we can write:

$$\rho_{\text{comp}}^2 \geq \min_{ij} \frac{p_{ij}^2}{2\left(\sigma_i^{-1}+L_{ij}^{-1}\right)} \frac{\sigma\kappa_i}{\lambda_{\min}^+(L)\, L_{ij}} \frac{\lambda_{\min}^+(L)}{\sigma\kappa}, \tag{3.C.5}$$

because $e_{ij}^T A^\dagger A e_{ij} = 1$ when (ij) is a virtual edge (because it is part of no cycle). Since $S_{\text{comp}} = \frac{1}{n}\sum_{i=1}^n \sum_{j=1}^m \sqrt{1+L_{ij}\sigma_i^{-1}}$, this can be rewritten:

$$\rho_{\text{comp}}^2 \geq \frac{r_\kappa}{2} \frac{p_{\text{comp}}^2}{n^2 S_{\text{comp}}^2}. \tag{3.C.6}$$

□

3.C.3. Execution time. Now that we have specified the rate of ADFS (improvement per iteration), we can bound the time needed to reach precision ε by plugging in the expected time to execute the schedule. In particular, we show in this section Theorem 19, which is a more precise version of Theorem 18.

We introduce Δ_p, r_κ and c_τ to quantify how heterogeneous the system is. More specifically, we can define $\sigma = \max_i \sigma_i$, $\kappa_i = 1 + \sigma_i^{-1}\sum_{j=1}^m L_{ij}$ and $\kappa_s = \max_i \kappa_i$. Since they are not all equal, we introduce $r_\kappa = \min_i \kappa_i/\kappa_s$. We choose the probabilities of virtual edges, such that $\sum_{j=1}^m p_{ij}$ is constant for all i and such that $p_{ij} = p_{\text{comp}}(1+L_{ij}\sigma_i^{-1})^{\frac{1}{2}}/(nS_{\text{comp}})$ for $S_{\text{comp}} = n^{-1}\sum_{i=1}^n\sum_{j=1}^m (1+L_{ij}\sigma_i^{-1})^{\frac{1}{2}}$. When (k,ℓ) is a communication edge, we further assume that $p_{k\ell} \geq \Delta_{\text{p}} p_{\text{comm}}/|E|$ for some constant $\Delta_{\text{p}} \leq 1$ and $p_{\text{comm}}^{\max} \leq c_\tau p_{\text{comm}}$ for some $c_\tau > 0$.

THEOREM 19. *We choose $\mu_{k\ell}^2 = \frac{1}{2}$ for communication edges, $\mu_{ij}^2 = \frac{\lambda_{\min}^+(L)}{\sigma\kappa_i} L_{ij}$ for computation edges and $p_{\text{comm}} = \min\left(\frac{1}{2}, \left(1+\sqrt{\frac{\tilde{\gamma}}{\kappa_{min}}} S_{\text{comp}}\right)^{-1}\right)$. Then, running Algorithm 4 for $K = \rho^{-1}\log(\varepsilon^{-1})$ iterations with the randomized sampling of Assumption 7 guarantees $\mathbb{E}\left[\|\theta_K - \theta^\star\|^2\right] \leq C_0\varepsilon$, and takes time $T(K)$, with $T(K)$ bounded by:*

$$T(K) \leq 2C\left(\frac{m+\sqrt{m\kappa_s}}{\sqrt{2r_\kappa}} + \frac{(1+4c_\tau\tau)}{\Delta_{\text{p}}}\sqrt{\frac{\kappa_s}{\tilde{\gamma}}}\right)\log\left(\frac{1}{\varepsilon}\right)$$

with probability tending to 1 as $\rho^{-1}\log(\varepsilon^{-1}) \to \infty$, where C is the same as in Theorem 17.

In heterogeneous settings, σ_i and sampling probabilities may be adapted to recover good guarantees, but this is beyond the scope of this paper. Note that taking computing probabilities exactly equal for all nodes is not necessary to ensure convergence, and only slightly slows down convergence. Indeed, it is always possible to analyze a schedule for which all

nodes have exactly the same probability of local update by adding a probability of doing nothing for time 1 as a local update to the nodes that are chosen less frequently. If we denote p_{wait} the probability that we need to add so that all nodes have the same probability of being selected, then $p_{\text{comp}} + p_{\text{comm}} = 1 - p_{\text{wait}}$ so θ_{comp} will be slightly smaller for a given p_{comm}. The actual algorithm can only be faster so this just gives a rough upper bound on the time to convergence.

PROOF. Using Theorem 17 on the average time per iteration, we know that as long as $p_{\text{comp}} > p_{\text{comm}}$, the execution time of the algorithm verifies the following bound for some $C > 0$ with high probability:

$$T(K) \leq \frac{C}{n} \left(p_{\text{comp}} + 2\tau p_{\text{comm}}^{\max}\right) K \tag{3.C.7}$$

Algorithm 4 requires $-\log(1/\varepsilon)/\log(1-\rho)$ iterations to reach error ε. Using that $\log(1+x) \leq x$ for any $x > -1$, we get that using $K_\varepsilon = \log(1/\varepsilon)\rho^{-1}$ instead also guarantees to make error less than ε. We now optimize the bound in ρ:

$$\frac{T(K_\varepsilon)}{\log\left(\varepsilon^{-1}\right)} \leq \frac{C}{n\rho} \left(p_{\text{comp}} + 2\tau p_{\text{comm}}^{\max}\right) \tag{3.C.8}$$

If we rewrite this in terms of ρ_{comm} and ρ_{comp}, we obtain:

$$\frac{T(K_\varepsilon)}{\log\left(\varepsilon^{-1}\right)} \leq C \max\left(T_1(p_{\text{comm}}), T_2(p_{\text{comm}})\right) \tag{3.C.9}$$

with

$$T_1(p_{\text{comm}}) = \frac{1}{n\rho_{\text{comm}}}(p_{\text{comp}} + 2c_\tau \tau p_{\text{comm}}) = \frac{2}{\Delta_p}\left(2c_\tau\tau - 1 + \frac{1}{p_{\text{comm}}}\right)\sqrt{\frac{\kappa}{\tilde{\gamma}}} \tag{3.C.10}$$

and

$$T_2(p_{\text{comm}}) = S_{\text{comp}}\sqrt{\frac{2}{r_\kappa}}\left(\frac{1 + (2c_\tau\tau - 1)p_{\text{comm}}}{1 - p_{\text{comm}}}\right) = S_{\text{comp}}\sqrt{\frac{2}{r_\kappa}}\left(1 + 2\tau\frac{p_{\text{comm}}}{1 - p_{\text{comm}}}\right) \tag{3.C.11}$$

T_1 is a continuous decreasing function of p_{comm} with $T_1 \to \infty$ when $p_{\text{comm}} \to 0$. Similarly, T_2 is a continuous increasing function of p_{comm} such that $p_{\text{comm}} \to \infty$ when $p_{\text{comm}} \to 1$. Therefore, the best upper bound on the execution time is given by taking $p_{\text{comm}} = p^*$ where p^* is such that $T_1(p^*) = T_2(p^*)$ and so $\rho_{\text{comm}}(p^*) = \rho_{\text{comp}}(p^*)$.

$$\frac{T(K_\varepsilon)}{\log\left(\varepsilon^{-1}\right)} \leq C T_1(p^*) \tag{3.C.12}$$

Then, p^* can be found by finding the root in $]0,1[$ of a second degree polynomial. In particular, p^* is the solution of:

$$p_{\text{comp}}^2 = p_{\text{comm}}^2 \frac{\tilde{\gamma}\Delta_p^2}{2\kappa r_\kappa} S_{\text{comp}}^2 = (1 - p_{\text{comm}})^2 \tag{3.C.13}$$

which leads to $p^* = \left(1 + \sqrt{\frac{\tilde{\gamma}}{2\kappa_{\min}}}\Delta_p S_{\text{comp}}\right)^{-1}$.

$$\begin{aligned}\frac{T(K_\varepsilon)}{\log(\varepsilon^{-1})} &\leq 2\frac{C}{\Delta_p}\left(2c_\tau\tau - 1 + \frac{1}{p^*}\right)\sqrt{\frac{\kappa}{\tilde{\gamma}}} \\ &\leq 2C\left(2\tau\frac{c_\tau}{\Delta_p}\sqrt{\frac{\kappa}{\tilde{\gamma}}} + \frac{1}{\sqrt{2r_\kappa}}S_{\text{comp}}\right)\end{aligned}$$

Finally, we use the concavity of the square root to show that:

$$\begin{aligned}S_{\text{comp}} &= \frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{m}\sqrt{1 + L_{ij}\sigma_i^{-1}} \\ &\leq \frac{1}{n}\sum_{i=1}^{n} m\sqrt{\sum_{j=1}^{m}\frac{1}{m}\left(1 + L_{i,j}\sigma_i^{-1}\right)} \\ &\leq \frac{1}{n}\sum_{i=1}^{n} m\sqrt{1 + \frac{1}{m}(\kappa_i - 1)} \\ &\leq m + \sqrt{m\kappa}\end{aligned}$$

Yet, this analysis only works as long as $p^* \leq 1/2$. When this constraint is not respected, we know that: $\tilde{\gamma}S_{\text{comp}}^2 \leq 2\kappa r_\kappa$. In this case, we can simply choose $p_{\text{comp}} = p_{\text{comm}} = \frac{1}{2}$ and then $\rho_{\text{comm}} \leq \rho_{\text{comp}}$, so

$$\frac{T(K_\varepsilon)}{\log(\varepsilon^{-1})} \leq CT_1\left(\frac{1}{2}\right) = 2\frac{C}{\Delta_p}\left(1 + 2c_\tau\tau\right)\sqrt{\frac{\kappa}{\tilde{\gamma}}} \tag{3.C.14}$$

The sum of the two bounds is a valid upper bound in all situations, which finishes the proof. □

CHAPTER 4

Dual-Free Decentralized Algorithm with Variance Reduction

For finite-sum problems, fast single-machine algorithms for large datasets rely on stochastic updates combined with variance reduction. Yet, existing decentralized stochastic algorithms either do not obtain the full speedup allowed by stochastic updates, or require oracles that are more expensive than regular gradients. In this Chapter, we introduce a Decentralized stochastic algorithm with Variance Reduction called DVR. DVR only requires computing stochastic gradients of the local functions, and is computationally as fast as a standard stochastic variance-reduced algorithms run on a $1/n$ fraction of the dataset, where n is the number of nodes. To derive DVR, we use Bregman coordinate descent on a well-chosen dual problem, and obtain a dual-free algorithm using a specific Bregman divergence. We give an accelerated version of DVR based on the Catalyst framework, and illustrate its effectiveness with simulations on real data.

This Chapter is based on the paper *Dual-Free Stochastic Decentralized Optimization with Variance Reduction* [Hendrikx, Bach, and Massoulié, 2020a], published at NeurIPS 2020.

4.1. Introduction

We consider the regularized empirical risk minimization problem distributed on a network of n nodes. Each node has a local dataset of size m, and the problem thus writes:

$$\min_{x\in\mathbb{R}^d} F(x) \triangleq \sum_{i=1}^{n} f_i(x), \text{ with } f_i(x) \triangleq \frac{\sigma_i}{2}\|x\|^2 + \sum_{j=1}^{m} f_{ij}(x), \tag{4.1.1}$$

where f_{ij} typically corresponds to the loss function for training example j of machine i, and σ_i is the local regularization parameter for node i. We assume that each function f_{ij} is convex and L_{ij}-smooth (see, *e.g.*, Nesterov [2013c]), and that each function f_i is M_i-smooth. Following Xiao et al. [2019b], we denote $\kappa_i = (1 + \sum_{i=1}^{m} L_{ij})/\sigma_i$ the stochastic condition number of f_i, and $\kappa_s = \max_i \kappa_i$. Similarly, the batch condition number is $\kappa_b = \max_i M_i/\sigma_i$. It always holds that $\kappa_b \leq \kappa_s \leq m\kappa_b$, but generally $\kappa_s \ll m\kappa_b$, which explains the success of stochastic methods. Indeed, $\kappa_s \approx m\kappa_b$ when all Hessians are orthogonal to one another which is rarely the case in practice, especially for a large dataset.

Regarding the distributed aspect, we follow the standard *gossip* framework [Boyd et al., 2006, Nedic and Ozdaglar, 2009, Duchi et al., 2012b, Scaman et al., 2017b] and assume that nodes are linked by a communication network which we represent as an undirected graph G. We denote $\mathcal{N}(i)$ the set of neighbors of node i and $\mathbb{1} \in \mathbb{R}^d$ the vector with all coordinates equal to 1. Communication is abstracted by multiplication by a positive semi-definite matrix $W \in \mathbb{R}^{n\times n}$, which is such that $W_{k\ell} = 0$ if $k \notin \mathcal{N}(\ell)$, and $\mathrm{Ker}(W) = \mathrm{Span}(\mathbb{1})$. The matrix W is called the *gossip matrix*, and we denote its spectral gap by $\gamma = \lambda_{\min}^{+}(W)/\lambda_{\max}(W)$, the ratio between the smallest non-zero and the highest eigenvalue of W, which is a key quantity

in decentralized optimization. We finally assume that nodes can compute a local stochastic gradient ∇f_{ij} in time 1, and that communication (*i.e.*, multiplication by W) takes time τ.

Single-machine stochastic methods. Problem (4.1.1) is generally solved using first-order methods. When m is large, computing ∇F becomes very expensive, and batch methods require $O(\kappa_b \log(\varepsilon^{-1}))$ iterations, which takes time $O(m\kappa_b \log(\varepsilon^{-1}))$, to minimize F up to precision ε. In this case, updates using the stochastic gradients ∇f_{ij}, where (i, j) is selected randomly, can be much more effective [Bottou, 2010]. Yet, these updates are noisy and plain stochastic gradient descent (SGD) does not converge to the exact solution unless the step-size goes to zero, which slows down the algorithm. One way to fix this problem is to use variance-reduced methods such as SAG [Schmidt et al., 2017], SDCA [Shalev-Shwartz and Zhang, 2013], SVRG [Johnson and Zhang, 2013b] or SAGA [Defazio et al., 2014a]. These methods require $O((nm + \kappa_s) \log(\varepsilon^{-1}))$ stochastic gradient evaluations, which can be much smaller than $O(m\kappa_b \log(\varepsilon^{-1}))$.

Decentralized methods. Decentralized adaptations of gradient descent in the smooth and strongly convex setting include EXTRA [Shi et al., 2015a], DIGing [Nedic et al., 2017] or NIDS [Li et al., 2019]. These algorithms have sparked a lot of interest, and the latest convergence results [Jakovetić, 2018, Xu et al., 2020b, Li and Lin, 2020] show that EXTRA and NIDS require time $O((\kappa_b + \gamma^{-1})(m + \tau)) \log(\varepsilon^{-1}))$ to reach precision ε. A generic acceleration of EXTRA using Catalyst [Li and Lin, 2020] obtains the (batch) optimal $O(\sqrt{\kappa_b}(1+\tau/\sqrt{\gamma}) \log(\varepsilon^{-1}))$ rate up to log factors. Another line of work on decentralized algorithms is based on the *penalty method* [Li et al., 2018, Dvinskikh and Gasnikov, 2019]. This consists in performing traditional optimization algorithms to problems augmented with a Laplacian penalty, and in particular enables the use of accelerated methods. Yet, these algorithms are sensitive to the value of the penalty parameter (when it is fixed), since it directly influences the solution they converge to. Another natural way to construct decentralized optimization algorithms is through dual approaches [Scaman et al., 2017b, Uribe et al., 2020]. Although the dual approach leads to algorithms that are optimal both in terms of number of communications and computations [Scaman et al., 2019, Hendrikx et al., 2020b], they generally assume access to the proximal operator or the gradient of the Fenchel conjugate of the local functions, which is not very practical in general since it requires solving a subproblem at each step.

Decentralized stochastic optimization. Although both stochastic and decentralized methods have a rich litterature, there exist few decentralized stochastic methods with linear convergence rate. Although DSA [Mokhtari and Ribeiro, 2016], or GT-SAGA [Xin et al., 2020a] propose such algorithms, they respectively take time $O((m\kappa_s + \kappa_s^4\gamma^{-1}(1+\tau) \log(\varepsilon^{-1}))$ and $O((m + \kappa_s^2\gamma^{-2})(1 + \tau) \log(\varepsilon^{-1}))$ to reach precision ε. Therefore, they have significantly worse rates than decentralized batch methods when $m = 1$, and than single-machine stochastic methods when $n = 1$. Other methods have better rates of convergence [Shen et al., 2018, Hendrikx et al., 2019b] but they require evaluation of proximal operators, which may be expensive.

Our contributions. This work develops a dual approach similar to that of Hendrikx et al. [2019b], which leads to a decentralized stochastic algorithm with rate $O(m + \kappa_s + \tau\kappa_b/\sqrt{\gamma})$, where the $\sqrt{\gamma}$ factor comes from Chebyshev acceleration, such as used in Scaman et al. [2017b]. Yet, our algorithm, called DVR, can be formulated in the primal only, thus avoiding the need for computing expensive dual gradients or proximal operators. Besides,

DVR is derived by applying Bregman coordinate descent to the dual of a specific augmented problem. Thus, its convergence follows from the convergence of block coordinate descent with Bregman gradients, which we prove as a side contribution. When executed on a single-machine, DVR is similar to dual-free SDCA [Shalev-Shwartz, 2016], and obtains similar rates. We believe that the same methodology could be applied to tackle non-convex problems, but we leave these extensions for future work.

We present in Section 4.2 the derivations leading to DVR, namely the dual approach and the dual-free trick. Then, Section 4.3 presents the actual algorithm along with a convergence theorem based on block Bregman coordinate descent (presented in Appendix 4.A). Section 4.4 shows how to accelerate DVR, both in terms of network dependence (Chebyshev acceleration) and global iteration complexity (Catalyst acceleration [Lin et al., 2017]). Finally, experiments on real-world data are presented in Section 4.5, that demonstrate the effectiveness of DVR.

4.2. Algorithm Design

This section presents the key steps leading to DVR. We start by introducing a relevant dual formulation from Hendrikx et al. [2019b], then introduce the dual-free trick based on Lan and Zhou [2017], and finally show how this leads to DVR, an actual implementable decentralized stochastic algorithm, as a special case of the previous derivations.

4.2.1. Dual formulation. The standard dual formulation of Problem (4.1.1) is obtained by associating a parameter vector to each node, and imposing that two neighboring nodes have the same parameters [Boyd et al., 2011, Jakovetić et al., 2014, Scaman et al., 2017b]. This leads to the following constrained problem, in which we write $\theta^{(i)} \in \mathbb{R}^d$ the local vector of node i:

$$\min_{\theta \in \mathbb{R}^{nd}} \sum_{i=1}^{n} f_i(\theta^{(i)}) \text{ such that } \forall k, \ell \in \mathcal{N}(k),\ \theta^{(k)} = \theta^{(\ell)}. \tag{4.2.1}$$

Following the approach of Hendrikx et al. [2019b, 2020b], we further split the $f_i(\theta^{(i)})$ term into $\sigma_i \|\theta^{(i)}\|^2/2 + \sum_{j=1}^{n} f_{ij}(\theta^{(ij)})$, with the constraint that $\theta^{(i)} = \theta^{(ij)}$ for all j. This is equivalent to the previous approach performed on an augmented graph [Hendrikx et al., 2019b, 2020b] in which each node is split into a star network with the regularization in the center and a local summand at each tip of the star. Thus, the equivalent augmented constrained problem that we consider writes:

$$\min_{\theta \in \mathbb{R}^{n(m+1)d}} \sum_{i=1}^{n} \left[\frac{\sigma_i}{2} \|\theta^{(i)}\|^2 + \sum_{j=1}^{m} f_{ij}(\theta^{(ij)}) \right] \text{s.t. } \forall k, \ell \in \mathcal{N}(k),\ \theta^{(k)} = \theta^{(\ell)} \text{ and } \forall i, j,\ \theta^{(i)} = \theta^{(ij)}. \tag{4.2.2}$$

We now use Lagrangian duality, and introduce two kinds of multipliers. The variable x corresponds to multipliers associated with the constraints given by edges of the communication graph (*i.e.*, $\theta^{(k)} = \theta^{(\ell)}$ if $k \in \mathcal{N}(\ell)$), that we will call *communication edges*. Similarly, y corresponds to the constraints associated with the edges that are specific to the augmented graph (*i.e.*, $\theta^{(i)} = \theta^{(ij)}\ \forall i, j$) that we call *computation* or *virtual edges*, since they are not present in the original graph and were constructed for the augmented problem. Therefore,

there are E communication edges (number of edges in the initial graph), and nm virtual edges. The dual formulation of Problem (4.2.2) thus writes:

$$\min_{x\in\mathbb{R}^{Ed},\ y\in\mathbb{R}^{nmd}} \frac{1}{2} q_A(x,y) + \sum_{i=1}^{n}\sum_{j=1}^{m} f_{ij}^*((A(x,y))^{(ij)}), \text{ with } q_A(x,y) \triangleq (x,y)^\top A^\top \Sigma A(x,y), \tag{4.2.3}$$

and where $(x,y) \in \mathbb{R}^{(E+nm)d}$ is the concatenation of vectors $x \in \mathbb{R}^{Ed}$, which is associated with the communication edges, and $y \in \mathbb{R}^{nmd}$, which is the vector associated with computation edges. We denote $\Sigma = \mathrm{Diag}(\sigma_1^{-1}, \cdots, \sigma_n^{-1}, 0, \cdots, 0) \otimes I_d \in \mathbb{R}^{n(m+1)d \times n(m+1)d}$ and A is such that for all $z \in \mathbb{R}^d$, $A(e_{k,\ell} \otimes z) = \mu_{k\ell}(u_k - u_\ell) \otimes P_{k\ell} z$ for edge (k,ℓ), where $P_{k\ell} = I_d$ if (k,ℓ) is a communication edge, P_{ij} is the projector on $\mathrm{Ker}(f_{ij})^\perp \triangleq (\cap_{x\in\mathbb{R}^d}\mathrm{Ker}(\nabla^2 f_{ij}(x)))^\perp$ if (i,j) is a virtual edge, $z_1 \otimes z_2$ is the Kronecker product of vectors z_1 and z_2, and $e_{k,\ell} \in \mathbb{R}^{E+nm}$ and $u_k \in R^{n(m+1)}$ are the unit vectors associated with edge (k,ℓ) and node k respectively.

Note that the upper left $nd \times nd$ block of $AA^\top$ (corresponding to the communication edges) is equal to $W \otimes I_d$ where W is a gossip matrix (see, *e.g.*, [Scaman et al., 2017b]) that depends on the $\mu_{k\ell}$. In particular, W is equal to the Laplacian of the communication graph if $\mu_{k\ell}^2 = 1/2$ for all (k,ℓ). For computation edges, the projectors P_{ij} account for the fact that the parameters $\theta^{(i)}$ and $\theta^{(ij)}$ only need to be equal on the subspaces on which f_{ij} is not constant, and we choose μ_{ij} such that $\mu_{ij}^2 = \alpha L_{ij}$ for some $\alpha > 0$. Although this introduces heavier notations, explicitly writing A as an $n(1+m)d \times (E+nm)d$ matrix instead of an $n(1+m) \times (E+nm)$ matrix allows to introduce the projectors P_{ij}, which then yields a better communication complexity than choosing $P_{ij} = I_d$. See Hendrikx et al. [2019b, 2020b] for more details on this dual formulation, and in particular on the construction on the augmented graph. Now that we have obtained a suitable dual problem, we would like to solve it without computing gradients or proximal operators of f_{ij}^*, which can be very expensive.

4.2.2. Dual-free trick. Dual methods are based on variants of Problem (4.2.3), and apply different algorithms to it. In particular, Scaman et al. [2017b], Uribe et al. [2020] use accelerated gradient descent [Nesterov, 2013c], and Hendrikx et al. [2019a,b] use accelerated (proximal) coordinate descent [Lin et al., 2015b]. Let p_{comm} denote the probability of performing a communication step and p_{ij} be the probability that node i samples a gradient of f_{ij}, which are such that for all i, $\sum_{j=1}^m p_{ij} = 1 - p_{\text{comm}}$. Applying a coordinate update with step-size η/p_{comm} to Problem (4.2.3) in the direction x (associated with communication edges) writes:

$$x_{t+1} = x_t - \eta p_{\text{comm}}^{-1} \nabla_x q_A(x_t, y_t), \tag{4.2.4}$$

where we denote ∇_x the gradient in coordinates that correspond to x (communication edges), and $\nabla_{y,ij}$ the gradient for coordinate (ij) (computation edge). Similarly, the standard coordinate update of a local computation edge (i,j) can be written as:

$$y_{t+1}^{(ij)} = \arg\min_{y\in\mathbb{R}^d} \left\{ \left(\nabla_{y,ij} q_A(x_t,y_t) + \mu_{ij}\nabla f_{ij}^*(\mu_{ij} y_t^{(ij)})\right)^\top y + \frac{p_{ij}}{2\eta}\|y - y_t^{(ij)}\|^2 \right\}, \tag{4.2.5}$$

where the minimization problem actually has a closed form solution. Yet, as mentioned before, solving Equation (4.2.5) requires computing the derivative of f_{ij}^*. In order to avoid this, a trick introduced by Lan and Zhou [2017] and later used in Wang and Xiao [2017] is to

replace the Euclidean distance term by a well-chosen Bregman divergence. More specifically, the Bregman divergence of a convex function ϕ is defined as:

$$D_\phi(x,y) = \phi(x) - \phi(y) - \nabla\phi(y)^\top(x-y). \tag{4.2.6}$$

Bregman gradient algorithms typically enjoy the same kind of guarantees as standard gradient algorithms, but with slightly different notions of *relative* smoothness and strong convexity [Bauschke et al., 2017, Lu et al., 2018]. Note that the Bregman divergence of the squared Euclidean norm is the squared Euclidean distance, and the standard gradient descent algorithm is recovered in that case. We now replace the Euclidean distance by the Bregman divergence induced by function $\phi : y \mapsto (L_{ij}/\mu_{ij}^2) f_{ij}^*(\mu_{ij} y^{(ij)})$, which is normalized to be 1-strongly convex since f_{ij}^* is L_{ij}^{-1}-strongly convex. We introduce the constant $\alpha > 0$ such that $\mu_{ij}^2 = \alpha L_{ij}$ for all computation edges (i,j). Using the definition of the Bregman divergence with respect to ϕ, we write:

$$\begin{aligned}
y_{t+1}^{(ij)} &= \arg\min_{y\in\mathbb{R}^d} \left(\nabla_{y,ij} q_A(x_t,y_t) + \mu_{ij}\nabla f_{ij}^*(\mu_{ij} y_t^{(ij)})\right)^\top y + \frac{p_{ij}}{\eta} D_\phi\left(y, y_t^{(ij)}\right) \\
&= \arg\min_{y\in\mathbb{R}} \left(\frac{\alpha\eta}{p_{ij}}\nabla_{y,ij} q_A(x_t,y_t) - \left(1 - \frac{\alpha\eta}{p_{ij}}\right)\mu_{ij}\nabla f_{ij}^*(\mu_{ij} y_t^{(ij)})\right)^\top y + f_{ij}^*(\mu_{ij} y) \\
&= \frac{1}{\mu_{ij}}\nabla f_{ij}\left(\left(1 - \frac{\alpha\eta}{p_{ij}}\right)\nabla f_{ij}^*(\mu_{ij} y_t^{(ij)}) - \frac{\alpha\eta}{\mu_{ij} p_{ij}}\nabla_{y,ij} q_A(x_t,y_t)\right).
\end{aligned}$$

In particular, if we know $\nabla f_{ij}^*(\mu_{ij} y_t^{(ij)})$ then it is possible to compute $y_{t+1}^{(ij)}$. Besides,

$$\nabla f_{ij}^*(\mu_{ij} y_{t+1}^{(ij)}) = (1-\alpha\eta)\nabla f_{ij}^*(\mu_{ij} y_t^{(ij)}) - \frac{\alpha\eta}{\mu_{ij}}\nabla_{y,ij} q_A(x_t,y_t), \tag{4.2.7}$$

so we can also compute $\nabla f_{ij}^*(\mu_{ij} y_{t+1}^{(ij)})$, and we can use it for the next step. Therefore, instead of computing a dual gradient at each step, we can simply choose $y_0^{(i)} = \mu_{ij}^{-1}\nabla f_{ij}(z_0^{(ij)})$ for any $z_0^{(ij)}$, and iterate from this. Therefore, the Bregman coordinate update applied to Problem (4.2.3) in the block of direction (i,j) with $y_0^{(ij)} = \mu_{ij}^{-1}\nabla f_i(z_0^{(ij)})$ yields:

$$z_{t+1}^{(ij)} = \left(1 - \frac{\alpha\eta}{p_{ij}}\right) z_t^{(ij)} - \frac{\alpha\eta}{p_{ij}\mu_{ij}}\nabla_{y,ij} q_A(x_t,y_t), \qquad y_{t+1}^{(ij)} = \mu_{ij}^{-1}\nabla f_i(z_{t+1}^{(ij)}). \tag{4.2.8}$$

The iterations of (4.2.8) are called a *dual-free* algorithm because they are a transformation of the iterations from (4.2.5) that do not require computing ∇f_{ij}^* anymore. This is obtained by replacing the Euclidean distance in (4.2.5) by the Bregman divergence of a function proportional to f_{ij}^*. Note that although we use the same dual-free trick the tools are different since Lan and Zhou [2017] applies a randomized primal-dual algorithm with fixed Bregman divergences choice to a specific *primal-dual* formulation. Instead, we apply a generic randomized Bregman coordinate descent algorithm to a specific *dual* formulation.

4.2.3. Distributed implementation. Iterations from (4.2.8) do not involve functions f_{ij}^* anymore, which was our first goal. Yet, they consist in updating dual variables associated with edges of the augmented graph, and have no clear distributed meaning yet. In this section, we rewrite the updates of (4.2.8) in order to have an easy to implement distributed algorithm. The key steps are (i) multiplication of the updates by A, (ii) expliciting the gossip

matrix and (iii) remarking that $\theta_t^{(i)} = (\Sigma A(x_t, y_t))^{(i)}$ converges to the primal solution for all i. For a vector $z \in \mathbb{R}^{(n+nm)d}$, we denote $[z]_{\text{comm}} \in \mathbb{R}^{nd}$ its restriction to the communication nodes, and $[M]_{\text{comm}} \in \mathbb{R}^{nd \times nd}$ similarly refers to the restriction on communication edges of a matrix $M \in \mathbb{R}^{(n+nm)d \times (n+nm)d}$. By abuse of notations, we call $A_{\text{comm}} \in \mathbb{R}^{nd \times Ed}$ the restriction of A to communication nodes and edges. We denote P_{comm} the projector on communication edges, and P_{comp} the projector on y. We multiply the x (communication) update in (4.2.8) by A on the left (which is standard [Scaman et al., 2017b, Hendrikx et al., 2019b]) and obtain:

$$A_{\text{comm}} x_{t+1} = A_{\text{comm}} x_t - \eta p_{\text{comm}}^{-1} [A P_{\text{comm}} A^\top]_{\text{comm}} [\Sigma A(x_t, y_t)]_{\text{comm}}. \tag{4.2.9}$$

Note that $[P_{\text{comm}} A^\top \Sigma A(x_t, y_t)]_{\text{comm}} = [P_{\text{comm}} A^\top]_{\text{comm}} [\Sigma A(x_t, y_t)]_{\text{comm}}$ because P_{comm} and Σ are non-zero only for communication edges and nodes. Similarly, and as previously stated, one can verify that $A_{\text{comm}} [P_{\text{comm}} A^\top]_{\text{comm}} = [A P_{\text{comm}} A^\top]_{\text{comm}} = W \otimes I_d \in \mathbb{R}^{nd \times nd}$ where W is a gossip matrix. We finally introduce $\tilde{x}_t \in \mathbb{R}^{nd}$ which is a variable associated with nodes, and which is such that $\tilde{x}_t = A_{\text{comm}} x_t$. With this rewriting, the communication update becomes:

$$\tilde{x}_{t+1} = \tilde{x}_t - \eta p_{\text{comm}}^{-1} (W \otimes I_d) \Sigma_{\text{comm}} \left[A(x_t, y_t)\right]_{\text{comm}}.$$

To show that $[A(x_t, y_t)]_{\text{comm}}$ is locally accessible to each node, we write:

$$[A(x_t, y_t)]_{\text{comm}}^{(i)} = (A_{\text{comm}} x_t)^{(i)} - \left(\sum_{k=1}^{n} \sum_{j=1}^{m} (A(e_{kj} \otimes y_t^{(kj)}))^{(i)} \right) = (\tilde{x}_t)^{(i)} - \sum_{j=1}^{m} \mu_{ij} y_t^{(ij)}.$$

We note this rescaled local vector $\theta_t = \Sigma_{\text{comm}}([A(x_t, y_t)]_{\text{comm}})$, and obtain for variables $\tilde{x}_t$ the gossip update of (4.2.11). Note that we directly write $y_t^{(ij)}$ instead of $P_{ij} y_t^{(ij)}$ even though there has been a multiplication by the matrix A. This is allowed because Equation (4.2.12) implies that (i) $y_t^{(ij)} \in \text{Ker}(f_{ij})^\perp$ for all t, and (ii) the value of $(I_d - P_{ij}) z_t^{(ij)}$ does not matter since $z_t^{(ij)}$ is only used to compute ∇f_{ij}. We now consider computation edges, and remark that:

$$\nabla_{y,ij} q_A(x_t, y_t) = -\mu_{ij} (\Sigma_{\text{comm}})_{ii} ([A(x_t, y_t)]_{\text{comm}})^{(i)} = -\mu_{ij} \theta_t. \tag{4.2.10}$$

Plugging Equation (4.2.10) into the updates of (4.2.8), we obtain the following updates:

$$\tilde{x}_{t+1} = \tilde{x}_t - \frac{\eta}{p_{\text{comm}}} (W \otimes I_d) \theta_t, \tag{4.2.11}$$

for communication edges, and for the local update of the j-th component of node i:

$$z_{t+1}^{(ij)} = \left(1 - \frac{\alpha\eta}{p_{ij}}\right) z_t^{(ij)} + \frac{\alpha\eta}{p_{ij}} \theta_t^{(i)}, \qquad \theta_{t+1}^{(i)} = \frac{1}{\sigma_i} \left(\tilde{x}_{t+1}^{(i)} - \sum_{j=1}^{m} \nabla f_{ij}(z_{t+1}^{(ij)}) \right). \tag{4.2.12}$$

Finally, Algorithm 8 is obtained by expressing everything in terms of θ_t and removing variable $\tilde{x}_t$. To simplify notations, we further consider θ as a matrix in $\mathbb{R}^{n \times d}$ (instead of a vector in $\mathbb{R}^{nd}$), and so the communication update of Equation (4.2.11) is a standard gossip update with matrix W, which we recall is such that $W \otimes I_d = [A P_{\text{comm}} A^\top]_{\text{comm}}$. We now discuss the local updates of Equation (4.2.12) more in details, which are closely related to dual-free SDCA updates [Shalev-Shwartz, 2016].

Algorithm 8 DVR(z_0)

$\alpha = 2\lambda_{\min}^{+}(A_{\text{comm}}^{\top} D_M^{-1} A_{\text{comm}})$, $\eta = \min\left(\frac{p_{\text{comm}}}{\lambda_{\max}(A_{\text{comm}}^{\top}\Sigma_{\text{comm}}A_{\text{comm}})}, \frac{p_{ij}}{\alpha(1+\sigma_i^{-1}L_{ij})}\right)$ *// Init.*
$\theta_0^{(i)} = -(\sum_{j=1}^{m} \nabla f_{ij}(z_0^{(ij)}))/\sigma_i$. *// z_0 is arbitrary but not θ_0.*
for $t = 0$ to $K-1$ **do** *// Run for K iterations*
  Sample u_t uniformly in $[0,1]$. *// Randomly decide the kind of update*
  if $u_t \leq p_{\text{comm}}$ **then**
    $\theta_{t+1} = \theta_t - \frac{\eta}{p_{\text{comm}}}\Sigma W \theta_t$ *// Communication using W*
  else
    for $i = 1$ to n **do**
      Sample $j \in \{1, \cdots, m\}$ with probability p_{ij}.
      $z_{t+1}^{(ij')} = z_t^{(ij')}$ for $j \neq j'$ *// Only one virtual node is updated*
      $z_{t+1}^{(ij)} = \left(1 - \frac{\alpha\eta}{p_{ij}}\right) z_t^{(ij)} + \frac{\alpha\eta}{p_{ij}}\theta_t^{(i)}$ *// Virtual node update*
      $\theta_{t+1}^{(i)} = \theta_t^{(i)} - \frac{1}{\sigma_i}\left(\nabla f_{ij}(z_{t+1}^{(ij)}) - \nabla f_{ij}(z_t^{(ij)})\right)$ *// Local update using f_{ij}*
return θ_K

4.3. Convergence Rate

The goal of this section is to set parameters η and α in order to get the best convergence guarantees. We introduce $\kappa_{\text{comm}} = \gamma\lambda_{\max}(A_{\text{comm}}^{\top}\Sigma_{\text{comm}}A_{\text{comm}})/\lambda_{\min}^{+}(A_{\text{comm}}^{\top}D_M^{-1}A_{\text{comm}})$, where $\lambda_{\min}^{+}$ and $\lambda_{\max}$ respectively refer to the smallest non-zero and the highest eigenvalue of the corresponding matrices. We denote D_M the diagonal matrix such that $(D_M)_{ii} = \sigma_i + \lambda_{\max}(\sum_{j=1}^{m} L_{ij}P_{ij})$, where $\nabla^2 f_{ij}(x) \preccurlyeq L_{ij}P_{ij}$ for all $x \in \mathbb{R}^d$. Note that we use notation κ_{comm} since it corresponds to a condition number. In particular, $\kappa_{\text{comm}} \leq \kappa_s$ when $\sigma_i = \sigma_j$ for all i, j, and κ_{comm} more finely captures the interplay between regularity of local functions (through D_M and Σ_{comm}) and the topology of the network (through A) otherwise.

THEOREM 20. *We choose $p_{\text{comm}} = \left(1+\gamma\frac{m+\kappa_s}{\kappa_{\text{comm}}}\right)^{-1}$, $p_{ij} \propto (1-p_{\text{comm}})(1+L_{ij}/\sigma_i)$ and α and η as in Algorithm 8. Then, there exists $C_0 > 0$ that only depends on θ_0 (initial conditions) such that for all $t > 0$, the error and the expected time T_ε required to reach precision ε are such that:*

$$\sum_{i=1}^{n} \frac{1}{2}\mathbb{E}[\|\theta_t^{(i)} - \theta^\star\|^2] \leq C_0\left(1 - \frac{\alpha\eta}{2}\right)^t, \text{ and so } T_\varepsilon = O\left(\left[m + \kappa_s + \tau\frac{\kappa_{\text{comm}}}{\gamma}\right]\log\varepsilon^{-1}\right).$$

PROOF SKETCH. We have seen in Section 4.2 that DVR is obtained by applying Bregman coordinate descent on a well-chosen dual problem. Therefore, one of our key results consists in proving convergence rates for Bregman coordinate descent in the relatively smooth setting. Although a similar algorithm is analyzed in Hanzely and Richtárik [2018], we give sharper results in the case of arbitrary sampling of blocks, and tightly adapt to the separability structure. This is crucial to our analysis since the probabilities to sample a local gradient and to communicate can be vastly different. In order to ease the reading of the paper, we present these results for a general setting in Appendix 4.A, which is self-contained and which we believe to be of independent interest (beyond its application to decentralized optimization).

Then, Appendix 4.B focuses on the application to decentralized optimization. In particular, we recall the Equivalence between DVR and Bregman coordinate descent applied to the dual problem of Equation (4.2.3), and show that its structure is suited to the application of coordinate descent. Indeed, no two virtual edges adjacent to the same node are updated at the same time with our sampling. Then, we evaluate the relative smoothness and strong convexity constants of the augmented problem, which is rather challenging due to the complex structure of the dual problem. This allows to derive adequate values for parameters α and η. Finally, we choose p_{comm} in order to minimize the execution time of DVR. □

We would like to highlight the fact that the convergence theory of DVR decomposes nicely into several building blocks, and thus simple rates are obtained. This is not so usual for decentralized algorithms, for instance many follow-up papers were needed to obtain a tight convergence theory for EXTRA [Shi et al., 2015a, Jakovetić, 2018, Xu et al., 2020b, Li and Lin, 2020]. We now discuss the convergence rate of DVR more in details.

Computation complexity. The computation complexity of DVR is the same computation complexity as locally running a stochastic algorithm with variance reduction at each node. This is not surprising since, as we argue later, DVR can be understood as a decentralized version of an algorithm that is closely related to dual-free SDCA [Shalev-Shwartz, 2016]. Therefore, this improves the computation complexity of EXTRA from $O(m(\kappa_b+\gamma^{-1}))$ individual gradients to $O(m+\kappa_s)$, which is the expected improvement for stochastic variance-reduced algorithm. In comparison, GT-SAGA [Xin et al., 2020a], a recent decentralized stochastic algorithm, has a computation complexity of order $O(m+\kappa_s^2/\gamma^2)$, which is significantly worse than that of DVR, and generally worse than that of EXTRA as well.

Communication complexity. The communication complexity of DVR (*i.e.*, the number of communications, so the communication time is retrieved by multiplying by τ) is of order $O(\kappa_{\text{comm}}/\gamma)$, and can be improved to $O(\kappa_{\text{comm}}/\sqrt{\gamma})$ using Chebyshev acceleration (see Section 4.4). Yet, this is in general worse than the $O(\kappa_b+\gamma^{-1})$ communication complexity of EXTRA or NIDS, which can be interpreted as a partly accelerated communication complexity since the optimal dependence is $O(\sqrt{\kappa_b/\gamma})$ [Scaman et al., 2019], and $2\sqrt{\kappa_b/\gamma} = \kappa_b+\gamma^{-1}$ in the worst case ($\kappa_b = \gamma^{-1}$). Yet, stochastic updates are mainly intended to deal with cases in which the computation time dominates, and we show in the experimental section that DVR outperforms EXTRA and NIDS for a wide range of communication times τ (the computation complexity dominates roughly as long as $\tau < \sqrt{\gamma}(m+\kappa_s)/\kappa_{\text{comm}}$). Finally, the communication complexity of DVR is significantly lower than that of DSA and GT-SAGA, the primal decentralized stochastic alternatives presented in Section 4.1.

Homogeneous parameter choice. In the homogeneous case ($\sigma_i = \sigma_j$ for all i, j), choosing the optimal p_{comp} and p_{comm} described above leads to $\eta\lambda_{\max}(W) = \sigma p_{\text{comm}}$. Therefore, the communication update becomes $\theta_{t+1} = (I - W/\lambda_{\max}(W))\,\theta_t$, which is a gossip update with a standard step-size (independent of the optimization parameters). Similarly, $\alpha\eta(m+\kappa_s) = p_{\text{comp}}$, and so the step-size for the computation updates is independent of the network.

Links with SDCA. The single-machine version of Algorithm 8 ($n = 1$, $p_{\text{comm}} = 0$) is closely related to dual-free SDCA [Shalev-Shwartz, 2016]. The difference is in the stochastic gradient used: DVR uses $\nabla f_{ij}(z_t^{(ij)})$, where $z_t^{(ij)}$ is a convex combination of $\theta_k^{(i)}$ for $k < t$,

whereas dual-free SDCA uses $g_t^{(ij)}$, which is a convex combination of $\nabla f_{ij}(\theta_k^{(i)})$ for $k < t$. Both algorithms obtain the same rates.

Local synchrony. Instead of using the synchronous communications of Algorithm 8, it is possible to update edges one at a time, as in Hendrikx et al. [2019b]. This can be very efficient in heterogeneous settings (both in terms of computation and communication times) and similar convergence results can be obtained using the same framework, and we leave the details for future work.

4.4. Acceleration

We show in this section how to modify DVR to improve the convergence rate of Theorem 20.

Network acceleration. Algorithm 8 depends on γ^{-1}, also called the *mixing time* of the graph, which can be as high as $O(n^2)$ for a chain of length n [Mohar, 1997]. However, it is possible to improve this dependency to $\gamma^{-1/2}$ by using Chebyshev acceleration, as in Scaman et al. [2017b]. To do so, the first step is to choose a polynomial P of degree k and communicate with $P(W)$ instead of W. In terms of implementation, this comes down to performing k communication rounds instead of one, but this makes the algorithm depend on the spectral gap of $P(W)$. Then, the important fact is that there is a polynomial P_γ of degree $\lceil \gamma^{-1/2} \rceil$ such that the spectral gap of $P_\gamma(W)$ is of order 1. Each communication step with $P_\gamma(W)$ only takes time $\tau \deg(P_\gamma) = \tau \lceil \gamma^{-1/2} \rceil$, and so the communication term in Theorem 20 can be replaced by $\tau \kappa_{\text{comm}} \gamma^{-1/2}$, thus leading to *network acceleration*. The polynomial P_γ can for example be chosen as a Chebyshev polynomial, and we refer the interested reader to Scaman et al. [2017b] for more details. Finally, other polynomials yield even faster convergence when the graph topology is known [Berthier et al., 2020].

Catalyst acceleration. Catalyst [Lin et al., 2015a] is a generic framework that achieves acceleration by solving a sequence of subproblems. Because of space limitations, we only present the accelerated convergence rate without specifying the algorithm in the main text. Yet, only mild modifications to Algorithm 8 are required to obtain these rates, and the detailed derivations and proofs are presented in Appendix 4.C.

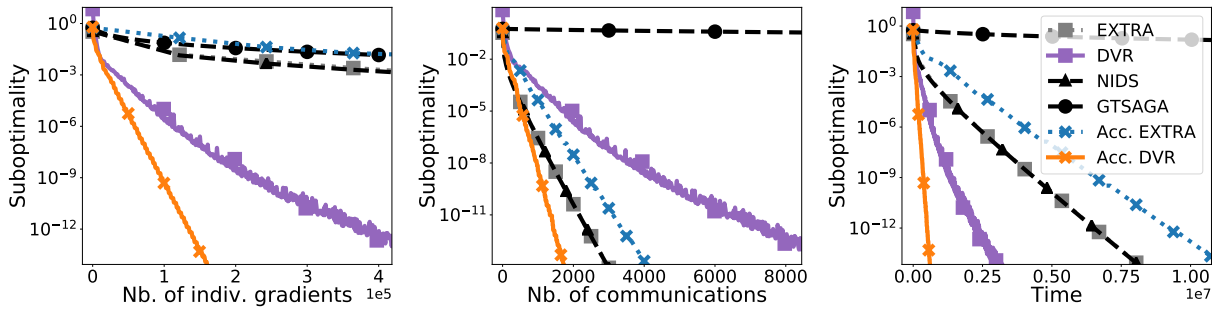
THEOREM 21. *DVR can be accelerated using catalyst, so that the time T_ε required to reach precision ε is equal (up to log factors) to*

$$T_\varepsilon = \tilde{O}\left(\left[m + \sqrt{m \kappa_s} + \tau \sqrt{\frac{\kappa_{\text{comm}}}{\gamma}} \times \sqrt{m \frac{\kappa_{\text{comm}}}{\kappa_s}} \right] \log \varepsilon^{-1} \right)$$

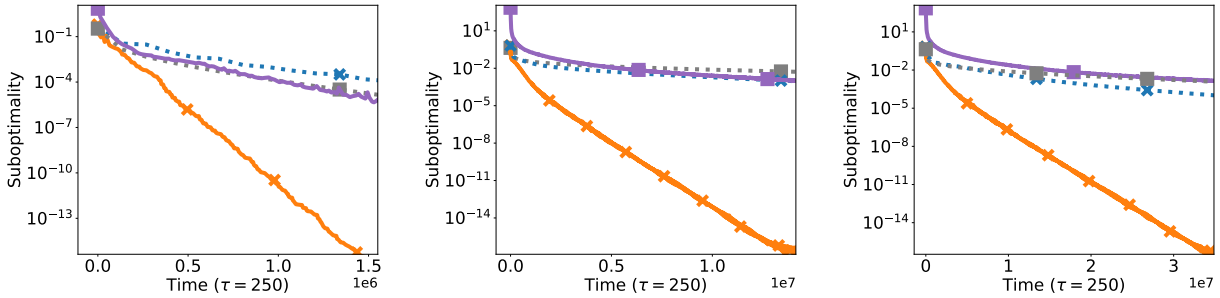
PROOF SKETCH. We follow the approach of Li and Lin [2020] to derive the algorithm, and apply Catalyst acceleration to the primal problem on the mean parameter $\bar{\theta}_t$ (which is never explicitly computed). Indeed, this conceptual algorithm can actually be implemented in a fully decentralized manner.

Then, we proceed to the actual proof, which requires a tight control over both primal and dual warm-start errors. Indeed, Theorem 23 (Appendix 4.B) controls dual variables but Catalyst acceleration is applied to the primal variables. □

This rate recovers the computation complexity of optimal finite sum algorithms such as ADFS [Hendrikx et al., 2019b, 2020b]. Although the communication time is slightly increased



(a) Erdős-Rényi, $\sigma = m \cdot 10^{-5}$



(b) Grid, $\sigma = m \cdot 10^{-5}$ (c) Erdős-Rényi, $\sigma = m \cdot 10^{-7}$ (d) Grid, $\sigma = m \cdot 10^{-7}$

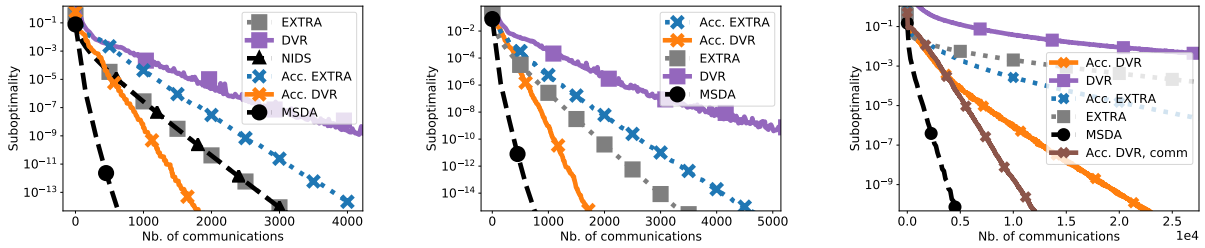
FIGURE 1. Experimental results for the RCV1 dataset with different graphs of size $n = 81$, with $m = 2430$ samples per node, and with different regularization parameters.

(by a factor $\sqrt{m\kappa_{\text{comm}}/\kappa_s}$), ADFS uses a stronger oracle than DVR (proximal operator instead of gradient), which is why we develop DVR in the first place. Although both ADFS and DVR are derived using the same dual formulation, both the approach and the resulting algorithms are rather different: ADFS uses accelerated coordinate descent, and thus has strong convergence guarantees at the cost of requiring dual oracles. DVR uses coordinate descent with the Bregman divergence of $\phi_{ij} \propto f_{ij}^*$ in order to work with primal oracles, but thus loses direct acceleration, which is recovered through the Catalyst framework. Note that the parameters of accelerated DVR can also be set such that $T_\varepsilon = \tilde{O}\left(\sqrt{\kappa_{\text{comm}}}\left[m + \tau/\sqrt{\gamma}\right]\log \varepsilon^{-1}\right)$, which recovers the convergence rate of optimal batch algorithms, but loses the finite-sum speedup.

4.5. Experiments

We investigate in this section the practical performances of DVR. We solve a regularized logistic regression problem on the RCV1 dataset [Lewis et al., 2004] ($d = 47236$) with $n = 81$ (leading to $m = 2430$) and two different graph topologies: an Erdős-Rényi random graph (see, *e.g.*, [Bollobás, 2001]) and a grid. We choose $\mu_{k\ell}^2 = 1/2$ for all communication edges, so the gossip matrix W is the Laplacian of the graph.

Figure 1 compares the performance of DVR with that of state-of-the-art primal algorithms such as EXTRA [Shi et al., 2015a], NIDS [Li et al., 2019], GT-SAGA [Xin et al.,



(a) Erdős-Rényi, $\sigma = m \cdot 10^{-5}$ (b) Grid, $\sigma = m \cdot 10^{-5}$ (c) Grid, $\sigma = m \cdot 10^{-7}$

FIGURE 2. Experimental results for the RCV1 dataset with different graphs of size $n = 81$, with $m = 2430$ samples per node, and with different regularization parameters.

2020a], and Catalyst accelerated versions of EXTRA [Li and Lin, 2020] and DVR. Suboptimality refers to $F(\theta_t^{(0)}) - F(\theta^\star)$, where node 0 is chosen arbitrarily and $F(\theta^\star)$ is approximated by the minimal error over all iterations. Each subplot of Figure 1(a) shows the same run with different x axes. The left plot measures the complexity in terms of individual gradients (∇f_{ij}) computed by each node whereas the center plot measures it in terms of communications (multiplications by W). All other plots are taken with respect to (simulated) time (*i.e.*, computing ∇f_{ij} takes time 1 and multiplying by W takes time τ) with $\tau = 250$ in order to report results that are independent of the computing cluster hardware and status. All parameters are chosen according to theory, except for the smoothness of the f_i, which requires finding the smallest eigenvalue of a $d \times d$ matrix. For this, we start with $L_b = \sigma_i + \sum_{j=1}^m L_{ij}$ (which is a known upper bound), and decrease it while convergence is ensured, leading to $\kappa_b = 0.01\kappa_s$. The parameters for accelerated EXTRA are chosen as in Li and Lin [2020] since tuning the number of inner iterations does not significantly improve the results (at the cost of a high tuning effort). For accelerated DVR, we set the number of inner iterations to N/p_{comp} (one pass over the local dataset). We use Chebyshev acceleration for (accelerated) DVR but not for (accelerated) EXTRA since it is actually slower, as predicted by the theory.

As expected from their theoretical iteration complexities, NIDS and EXTRA perform very similarly Li and Lin [2020], and GT-SAGA is the slowest method. Therefore, we only plot NIDS and GT-SAGA in Figure 1(a). We then see that though it requires more communications, DVR has a much lower computation complexity than EXTRA, which illustrates the benefits of stochastic methods. We see that DVR is faster overall if we choose $\tau = 250$, and both methods perform similarly for $\tau \approx 1000$, at which point communicating takes roughly as much time as computing a full local gradient. We then see that accelerated EXTRA has quite a lot of overhead and, despite our tuning efforts, is slower than EXTRA when the regularization is rather high. On the other hand, accelerated DVR consistently outperforms DVR by a relatively large margin. The communication complexity is in particular greatly improved, allowing accelerated DVR to be the fastest method regardless of the setting.

Finally, Figure 2 presents the comparison between DVR and MSDA [Scaman et al., 2017b], an optimal decentralized batch algorithm, in terms of communication complexity. To implement MSDA, we compute the dual gradients by solving each local subproblem ($\nabla f^*(x) = \arg\max_y x^\top y - f(y)$) up to precision 10^{-11} using accelerated gradient descent. Solving the subproblems with lower precision caused MSDA to plateau and not converge to

the true optimum. In Figure 2(c), *Acc. DVR comm* (the brown line) refers to Accelerated DVR with Catalyst parameter chosen to favor communication complexity (as explained after Theorem 21). MSDA is the fastest algorithm as expected, but accelerated DVR is not too far behind, especially given the fact that it relies on generic Catalyst acceleration, which adds some complexity overhead. Therefore, the comparison with MSDA corroborates the fact that accelerated DVR is competitive with optimal methods in terms of communication while enjoying a drastically lower computational cost. Further experimental results are given in Appendix 4.D, and the code is available in supplementary material and at `https://github.com/HadrienHx/DVR_NeurIPS`.

4.6. Conclusion

This paper introduces DVR, a Decentralized stochastic algorithm with Variance Reduction obtained using Bregman block coordinate descent on a well-chosen dual formulation. Thanks to this approach, DVR inherits from the fast rates and simple theory of dual approaches without the computational burden of relying on dual oracles. Therefore, DVR has a drastically lower computational cost than standard primal decentralized algorithms, although sometimes at the cost of a slight increase in communication complexity. The framework used to derive DVR is rather general and could in particular be extended to analyze asynchronous algorithms. Finally, although deriving a direct acceleration of DVR is a challenging open problem, Catalyst and Chebyshev accelerations allow to significantly reduce DVR's communication overhead both in theory and in practice.

This appendix contains the details of the derivations and proofs from the main text. More specifically, Appendix 4.A is a self-contained appendix that specifies the Bregman coordinate descent algorithm and proves its convergence rate. Appendix 4.B focuses on the application of Bregman coordinate descent to the dual problem (relative smoothness and strong convexity constants, sparsity structure), and how to retrieve guarantees on the primal parameters. Appendix 4.C is devoted to presenting the Catalyst acceleration of DVR and proving its convergence speed, and Appendix 4.D details the experimental setting, along with more experiments.

4.A. Bregman Coordinate Descent

We focus in this section on the general problem minimizing $f + g$ using coordinate Bregman gradient, where g is separable, *i.e.*, $g(x) = \sum_{i=1}^{d} g_i(x^{(i)})$. This is a self-contained section, and notations may differ from the rest of the paper. In particular, function f is for now arbitrary and not related to F or f_i from Problem (4.1.1), and the dimension d is arbitrary as well.

We first precise the blocks sampling rule. More specifically, we define a block $b \subset \{1, \dots, d\}$ as a collection of coordinates, and $\mathcal{B}$ is the set of all blocks that can be chosen for the updates. Then, the algorithm updates each block $b \in \mathcal{B}$ with probability $p(b)$, so that the probability of updating a given coordinate is given by $p_i = \sum_{i \in b} p(b)$. Similarly to individual coordinates, we write $x^{(b)}$ the restriction of x to coordinates in b. The Bregman coordinate gradient update for a block of coordinates b writes:

$$x_{t+1} = \arg\min_{x \in \mathbb{R}^d} \left\{ V_t^b(x) \triangleq \sum_{i \in b} \frac{\eta_t}{p_i} \left[\nabla_i f(x_t)^\top x + g_i(x^{(i)}) \right] + D_\phi(x, x_t) \right\}, \tag{4.A.1}$$

where $\nabla_i f$ denotes the gradient of f in direction i. In order to derive strong guarantees for this block coordinate descent algorithm, we need to ensure that there is some separability in functions f and ϕ, and that the block structure is suited to this separability. All the assumptions about the separability structure of f, g and ϕ are contained in the following assumption.

Assumption 8 (Separability). *The function g is separable and the function ϕ is block-separable for b, meaning that for all $b \in \mathcal{B}$, there exist two convex functions ϕ_b and $\phi_b^\perp$ such that for all x,*

$$\phi(x) = \phi_b(x^{(b)}) + \phi_b^\perp(x - x^{(b)}). \tag{4.A.2}$$

Besides, for all $b \in \mathcal{B}$, either of the following two hold:

(1) ϕ and f are separable for b, i.e., *$\phi_b(x^{(b)}) = \sum_{i \in b} \phi_i(x^{(i)})$, and*

$$\sum_{i \in b} \left[f(x_t + \delta_i e_i) - f(x_t) \right] = f\left(x_t + \sum_{i \in b} \delta_i e_i \right) - f(x_t).$$

(2) $p_i = p_j$ for all $i, j \in b$.

If ϕ is not block-separable, the support of the Bregman update in direction b may not restricted to b. This causes some of the derivations below to fail, which is why we prevent it by assuming that Equation (4.A.2) holds.

Then, the first option ensures that within a block, the updates do not affect each other. The function f is not separable, but some directions can be updated independently from others. To have these independent updates, we also need to assume further separability of ϕ within the blocks. The second option states that if only block-separability of ϕ is assumed then within each block for which ϕ and f are not separable, coordinates must be picked with the same probability.

Assumption 8 is a bit technical but we actually require all statements in Section 4.2. In particular, we will see that the first option is verified for *computation* updates, whereas the second option will be verified for *communication* updates.

Now that we have made assumptions on the structure of f, g and ϕ, we will make assumptions on their regularity. We start by a directional relative smoothness assumption between f and ϕ, *i.e.*, we assume that for all i, there exists L^i_{rel} such that for all $\delta > 0$ and e_i the unit vector of direction i,

$$D_f(x + \delta e_i, x) \leq L^i_{\text{rel}} D_\phi(x + \delta e_i, x). \tag{4.A.3}$$

Similarly, for $\sigma_{\text{rel}} > 0$, f is said to be σ_{rel}-strongly convex relatively to ϕ if for all x, y:

$$D_f(x, y) \geq \sigma_{\text{rel}} D_\phi(x, y). \tag{4.A.4}$$

We finally assume that f and ϕ are convex (but not necessarily smooth). We can now state the central theorem of this section:

THEOREM 22. *Let f and ϕ be such that f is L^i_{rel}-smooth in direction i and σ_{rel}-strongly convex relatively to ϕ. Denote $p_{\min} = \min_i p_i$, and*

$$L_t = D_\phi(x, x_t) + \frac{\eta_t}{p_{\min}} \left(F(x_t) - F(x)\right).$$

Then, if the blocks $\mathcal{B}$ respect Assumption 8 (separability) and $\eta_t L^i_{\text{rel}} < p_i$ for all i, the Bregman coordinate descent algorithm guarantees for all x:

$$\mathbb{E}[L_{t+1}] \leq (1 - \eta_t \sigma_{\text{rel}}) L_t.$$

The same result holds with $L'_t = D_\phi(x, x_t) + \frac{1}{L^{\max}_{\text{rel}}} \left(F(x_t) - F(x)\right)$, where $L^{\max}_{\text{rel}} = \max_i L^i_{\text{rel}}$.

Theorem 22 is a mix of the coordinate descent (Theorem 4) and Bregman gradient results (Theorem 5) presented in Section 1.1. To prove this theorem, we start by proving the monotonicity of such iterations.

LEMMA 17 (Monotonicity). *We note $\delta_i = e_i^\top (x_{t+1} - x_t) e_i$. If $x_{t+1} = \arg\min_x V^b_t(x)$ then:*

(1) If ϕ and f are separable for b then for all $i \in b$, if $\eta_t L^i_{\text{rel}} \leq p_i$ then $F(x_t) \geq F(x_t + \delta_i)$.
(2) If $p_i = p_j$ for all $i, j \in b$ and $\eta_t L^b_{\text{rel}} \leq p_b$ then $F(x_t) \geq F(x_{t+1})$.

PROOF. We start by the first point. If ϕ is separable for b then this means that each coordinate is updated independently. By definition of $x^{(i)}_{t+1}$, we have $V^b_t(x^{(b)}_{t+1}) \leq V^b_t(x_t)$. This writes, splitting over each i and using the fact that $D_{\phi_i}(x_t, x_t) = 0$:

$$\begin{aligned} g_i(x^{(i)}_t) - g_i(x^{(i)}_{t+1}) &\geq \nabla_i f(x_t)^\top (x^{(i)}_{t+1} - x^{(i)}_t) + \frac{p_i}{\eta} D_{\phi_i}(x^{(i)}_{t+1}, x^{(i)}_t) \\ &= \nabla_i f(x_t)^\top (x_t + \delta_i - x_t) + \frac{p_i}{\eta} D_{\phi_i}(x^{(i)}_{t+1}, x^{(i)}_t) \end{aligned}$$

$$
\begin{aligned}
&= f(x_t + \delta_i) - f(x_t) - D_f(x_{t+1}^{(i)}, x_t^{(i)}) + \frac{p_i}{\eta} D_{\phi_i}(x_{t+1}^{(i)}, x_t^{(i)}) \\
&\geq f(x_t + \delta_i) - f(x_t) + \left(\frac{p_i}{\eta} - L_{\text{rel}}^i \right) D_{\phi_i}(x_{t+1}^{(i)}, x_t^{(i)}) \\
&\geq f(x_t + \delta_i) - f(x_t).
\end{aligned}
$$

The result follows from summing over all $i \in b$, and using Assumption 8. For the second point, it is not possible to split the update per coordinate since ϕ is not separable. Yet, we can still write (using separability of g):

$$
\sum_{i \in b} \frac{\eta_t}{p_i} \left[g(x_t^{(i)}) - g(x_{t+1}^{(i)}) - \nabla_i f(x_t)^\top (x_{t+1}^{(i)} - x_t^{(i)}) \right] \geq D_\phi(x_{t+1}, x_t). \tag{4.A.5}
$$

Since g is separable and $p_i = p_b$ for all $i \in b$, Equation (4.A.5) writes:

$$
g(x_t) - g(x_{t+1}) \geq \nabla f(x_t)^\top (x_{t+1} - x_t) + \frac{p_b}{\eta} D_\phi(x_{t+1}, x_t). \tag{4.A.6}
$$

Note that this crucially relies on $x_{t+1} - x_t$ having support on b, which is enforced by the block-separability of ϕ. Then, the proof is similar to that of the first point, using that $\eta_t L_{\text{rel}}^b \leq p_b$. □

Using this monotonicity result allows us to prove Theorem 22.

Proof of Theorem 22. First note that by convexity of all g_i,

$$
\nabla^2 V_t^b(x) = \sum_{i \in b} \frac{\eta_t}{p_i} \nabla^2 g_i(x^{(i)}) + \nabla^2 \phi(x) \succcurlyeq \nabla^2 \phi(x).
$$

Therefore, we have $D_{V_t^b}(x, y) \geq D_\phi(x, y)$ for all $x, y \in \mathbb{R}^d$. Applying this with $y = x_{t+1}$ yields:

$$
V_t^b(x) - V_t^b(x_{t+1}) - \nabla V_t^b(x_{t+1})^\top (x - x_{t+1}) \geq D_\phi(x, x_{t+1}). \tag{4.A.7}
$$

Then, $\nabla V_t^b(x_{t+1}) = 0$ by definition of x_{t+1}, so Equation (4.A.7) writes:

$$
\begin{aligned}
D_\phi(x, x_{t+1}) + \sum_{i \in b} \frac{\eta_t}{p_i} \left(g_i(x_{t+1}^{(i)}) - g(x^{(i)}) \right) \leq \sum_{i \in b} \frac{\eta_t}{p_i} \nabla_i f(x_t)^\top (x - x_{t+1}) \\
+ D_\phi(x, x_t) - D_\phi(x_{t+1}, x_t).
\end{aligned}
$$

We first consider that the first option of Assumption 8 holds, *i.e.*, that f and ϕ are separable in b. We note $\delta_i = e_i^\top (x_{t+1} - x_t) e_i$, so that:

$$
\begin{aligned}
-\nabla_i f(x_t)^\top (x_{t+1} - x_t) &= \nabla f(x_t)^\top (x_t + \delta_i - x_t) \\
&= f(x_t) - f(x_t + \delta_i) + D_f(x_t + \delta_i, x_t) \\
&\leq f(x_t) - f(x_t + \delta_i) + L_{\text{rel}}^i D_{\phi_i}(x_{t+1}^{(i)}, x_t^{(i)}).
\end{aligned}
$$

Therefore, if $\eta_t L_{\text{rel}}^i \leq p_i$ for all $i \in b$,

$$
\begin{aligned}
&- \sum_{i \in b} \frac{\eta_t}{p_i} \nabla_i f(x_t)^\top (x_{t+1} - x_t) - D_\phi(x_{t+1}, x_t) \\
&\qquad \leq \sum_{i \in b} \frac{\eta_t}{p_i} \left[f(x_t) - f(x_t + \delta_i) \right] + \sum_{i \in b} \left(\frac{\eta_t L_{\text{rel}}^i}{p_i} - 1 \right) D_{\phi_i}(x_{t+1}^{(i)}, x_t^{(i)})
\end{aligned}
$$

$$\leq \sum_{i\in b} \frac{\eta_t}{p_i} \left[f(x_t) - f(x_t + \delta_i)\right]$$

The $g_i(x_{t+1}^{(i)}) - g_i(x^{(i)})$ term can be replaced by $g(x_t + \delta_i) - g(x_t) + g_i(x_t^{(i)}) - g_i(x^{(i)})$ since $g_j(x_{t+1}) = g_j(x_t)$ for $j \neq i$. Therefore, we obtain:

$$\begin{aligned} &D_\phi(x, x_{t+1}) + \sum_{i\in b} \frac{\eta_t}{p_i} \left[F(x_t + \delta_i) - F(x_t)\right] + \sum_{i\in b} \frac{\eta_t}{p_i} \left(g_i(x_t^{(i)}) - g_i(x^{(i)})\right) \\ &\leq \sum_{i\in b} \frac{\eta_t}{p_i} \nabla_i f(x_t)^\top (x - x_t) + D_\phi(x, x_t). \end{aligned} \tag{4.A.8}$$

The separability of F in b and its monotonicity lead to, using the fact that $x_{t+1} = x_t + \sum_{i\in b} \delta_i$:

$$\sum_{i\in b} \frac{\eta_t}{p_i} \left[F(x_t + \delta_i) - F(x_t)\right] \geq \frac{\eta_t}{p_{\min}} \sum_{i\in b} \left[F(x_t + \delta_i) - F(x_t)\right] = \frac{\eta_t}{p_{\min}} \left[F(x_{t+1}) - F(x_t)\right].$$

Therefore, if the first option of Assumption 8 holds, we obtain:

$$\begin{aligned} &D_\phi(x, x_{t+1}) + \frac{\eta_t}{p_{\min}} \left[F(x_{t+1}) - F(x_t)\right] + \sum_{i\in b} \frac{\eta_t}{p_i} \left(g_i(x_t^{(i)}) - g_i(x^{(i)})\right) \\ &\leq \sum_{i\in b} \frac{\eta_t}{p_i} \nabla_i f(x_t)^\top (x - x_t) + D_\phi(x, x_t). \end{aligned} \tag{4.A.9}$$

If the second option holds, *i.e.*, $p_i = p$ for all $i \in b$, then

$$\sum_{i\in b} \frac{\eta_t}{p_i} \nabla_i f(x_t)^\top (x_{t+1} - x_t) = \frac{\eta_t}{p} \nabla f(x_t)^\top (x_{t+1} - x_t),$$

and Equation (4.A.9) can be obtained through similar derivations (at the block-level). Using the separability of g, we obtain that

$$\mathbb{E}[\sum_{i\in b} \frac{1}{p_i} \left(g_i(x_t^{(i)}) - g_i(x^{(i)})\right)] = g(x_t) - g(x).$$

Then, since $\mathbb{E}[\sum_{i\in b} \frac{1}{p_i} \nabla_i f(x_t)] = \sum_i p_i^{-1} \sum_{b: i\in b} p(b) \nabla_i f(x_t) = \nabla f(x_t)$, and the relative strong convexity assumption yields:

$$\mathbb{E}[\sum_{i\in b} \frac{1}{p_i} \nabla_i f(x_t)^\top (x - x_t)] = \nabla f(x_t)^\top (x - x_t) \leq f(x) - f(x_t) - \sigma_{\text{rel}} D_\phi(x, x_t).$$

Therefore, taking the expectation of Equation (4.A.8) yields:

$$\mathbb{E}[D_\phi(x, x_{t+1}) + \frac{\eta_t}{p_{\min}} \left(F(x_{t+1}) - F(x_t)\right)] \leq \eta_t \left(F(x) - F(x_t)\right) + (1 - \eta_t \sigma_{\text{rel}}) D_\phi(x, x_t).$$

We obtain after some rewriting:

$$\begin{aligned} &\mathbb{E}[D_\phi(x, x_{t+1}) + \frac{\eta_t}{p_{\min}} \left(F(x_{t+1}) - F(x)\right)] \\ &\leq (1 - p_{\min}) \frac{\eta_t}{p_{\min}} \left(F(x_t) - F(x)\right) + (1 - \eta_t \sigma_{\text{rel}}) D_\phi(x, x_t). \end{aligned}$$

Finally, $\sigma_{\text{rel}} \leq L_{\text{rel}}^i$ so $\eta_t \sigma_{\text{rel}} \leq \eta_t L_{\text{rel}}^i \leq p_i$ for all i, and in particular $1 - p_{\min} \leq 1 - \eta_t \sigma_{\text{rel}}$, which yields the desired result.

The result on L_t' is be obtained by bounding $\eta / p_{\min}$ by $L_{\text{rel}}^{\max} = \max_i L_{\text{rel}}^i$ and remarking that $1 - \eta_t L_{\text{rel}}^{\max} \leq 1 - \eta_t \sigma_{\text{rel}}$ since $L_{\text{rel}}^{\max} \geq \sigma_{\text{rel}}$. □

4.B. Convergence results for DVR

We now give a series of small results, that justify our approach. We start by showing the applicability of Theorem 22 to Problem (4.2.3), and the associated constants. Finally, we show how to obtain rates for the primal iterates θ_t.

4.B.1. Application to the dual of the augmented problem. In this section, we note $f^*_{\text{sum}} = \sum_{i=1}^n \sum_{j=1}^m f^*_{ij}$, so that Problem 4.2.3 writes:

$$\min_{x,y} q_A(x,y) + f^*_{\text{sum}}(y) \tag{4.B.1}$$

LEMMA 18. *The iterations of Algorithm 8 are equivalent to the iteration of Equations (4.A.1) applied to Problem (4.2.3) with $g = 0$ and $\phi(x,y) = \phi_{\text{comm}}(x) + \sum_{i=1}^n \sum_{j=1}^m \phi_{ij}(y^{(ij)})$, with $\phi_{\text{comm}}(x) = \frac{1}{2}\|x\|^2_{A^\dagger A}$ for coordinates associated with communication edges, and $\phi_{ij}(y^{(ij)}) = \frac{L_{ij}}{\mu_{ij}^2} f^*_{ij}(\mu_{ij} y_{ij})$ for coordinates associated with computation edges.*

PROOF. This result follows from the dual-free and implementation-friendly derivations presented in the previous section. □

LEMMA 19. *Let $\alpha = 2\lambda_{\min}(A_{\text{comm}}^\top D_M^{-1} A_{\text{comm}})$, and ϕ as in Lemma 18, then:*

(1) $q_A + f^*_{\text{sum}}$ is $(\alpha/2)$-strongly convex relatively to ϕ.

(2) $q_A + f^*_{\text{sum}}$ is $(L^{\text{comm}}_{\text{rel}})$-smooth relatively to ϕ in the direction of communication edges, with

$$L^{\text{comm}}_{\text{rel}} = \lambda_{\max}(A_{\text{comm}}^\top \Sigma_{\text{comm}} A_{\text{comm}}).$$

(3) $q_A + f^*_{\text{sum}}$ is (L^{ij}_{rel})-smooth relatively to ϕ in the direction of virtual edge (i,j), with

$$L^{ij}_{\text{rel}} = \alpha\left(1 + \frac{L_{ij}}{\sigma_i}\right).$$

PROOF. First note that $\nabla^2 f^*_{\text{sum}}$ is a block-diagonal matrix, and its ij-th block is equal to

$$(\nabla^2 f^*_{\text{sum}}(y))_{ij} = A^\top (u_{ij}u_{ij}^\top \otimes \nabla^2 f^*_{ij}(\mu_{ij} y^{(ij)}))A \succcurlyeq \frac{1}{L_{ij}} A^\top (u_{ij}u_{ij}^\top \otimes P_{ij})A, \tag{4.B.2}$$

where $u_{ij} \in \mathbb{R}^{n(1+m)}$ denotes the unit vector corresponding to virtual *node* (i,j). We denote $\tilde{\Sigma} = \Sigma + \sum_{i=1}^n \sum_{j=1}^m \frac{1}{L_{ij}}(u_{ij}u_{ij}^\top) \otimes I_d$. Then,

$$\nabla^2 q_A(x,y) + \nabla^2 f^*_{\text{sum}}(y) = A^\top \tilde{\Sigma} A + \nabla^2 f^*_{\text{sum}}(y) - A^\top \left[\sum_{i=1}^n \sum_{j=1}^m \frac{1}{L_{ij}}(u_{ij}u_{ij}^\top) \otimes P_{ij}\right] A. \tag{4.B.3}$$

Relative strong convexity.. Then, Hendrikx et al. [2020b, Lemma 6.5] leads to $A^\top \tilde{\Sigma} A \succcurlyeq \sigma_F A^\dagger A$. Note that the notations are slightly different, and the matrix $\tilde{\Sigma}$ in this paper is the same as the matrix $\Sigma^\dagger$ in Hendrikx et al. [2020b]. Then, remark that $(A^\dagger A)_{ij} = P_{ij} = \frac{1}{\mu_{ij}^2}(A^\top[(u_{ij}u_{ij}^\top) \otimes P_{ij}]A)_{ij}$, and $\phi_{ij} = \alpha^{-1} f^*_{ij}$, so that:

$$\nabla^2 q_A(x,y) + \nabla^2 f^*_{\text{sum}}(y) \succcurlyeq \sigma_F \nabla^2 \phi(x,y) + (1 - \alpha^{-1}\sigma_F)\left[\nabla^2 f^*_{\text{sum}}(y) - A^\top \left[\sum_{i=1}^n \sum_{j=1}^m \frac{1}{L_{ij}}(u_{ij}u_{ij}^\top) \otimes P_{ij}\right] A.\right].$$

Finally, using that Equation (4.B.2) along with the fact that $\sigma_F \leq \alpha$ implies that $q_A + f^*_{\text{sum}}$ is σ_F-relatively strongly convex with respect to ϕ.

Relative smoothness.. We first prove the relative smoothness property for communicate edges. For any $\tilde{x} \in R^{Ed}$, Equation (4.B.3) leads to:

$$(\tilde{x}, 0)^\top [\nabla^2 q_A(x, y) + \nabla^2 f^*_{\text{sum}}(y)](\tilde{x}, 0) = (\tilde{x}, 0)^\top A^\top \Sigma A (\tilde{x}, 0) \preccurlyeq L^{\text{comm}}_{\text{rel}} (\tilde{x}, 0)^\top \nabla^2 \phi(x, y)(\tilde{x}, 0).$$

Similarly, for any $\theta \in \mathbb{R}^d$, we consider $\tilde{y} = e_{ij} \otimes \theta$ and write:

$$\begin{aligned}\tilde{y}^\top [\nabla^2 q_A(x, y) + \nabla^2 f^*_{\text{sum}}(y)]\tilde{y} &= \tilde{y}^\top A^\top \tilde{\Sigma} A \tilde{y} + \mu_{ij}^2 \theta^\top \left[\nabla^2 f^*_{ij}(\mu_{ij} y^{(ij)}) - \frac{1}{L_{ij}} P_{ij}\right] \theta \\ &\preccurlyeq L^i_{\text{rel}} \tilde{y}^\top \nabla^2 \phi(x, y) \tilde{y} + (1 - \alpha^{-1} L^i_{\text{rel}}) \theta^\top \left[\nabla^2 f^*_{ij}(\mu_{ij} y^{(ij)}) - \frac{1}{L_{ij}} P_{ij}\right] \theta,\end{aligned}$$

with

$$L^i_{\text{rel}} = \max_\theta \mu_{ij}^2 u_{ij}^\top \tilde{\Sigma} u_{ij} \frac{\theta^\top P_{ij} \theta}{\|\theta\|^2} \leq \alpha \left(1 + \frac{L_{ij}}{\sigma_i}\right).$$

Finally, $\nabla^2 f^*_{ij}(\mu_{ij} y^{(ij)}) \succcurlyeq P_{ij}/L_{ij}$, and $\alpha \leq L^i_{\text{rel}}$, which ends the proof of the directional relative smoothness result. □

LEMMA 20. *Assumption 8 holds with $f = q_A + f^*_{\text{sum}}$, $g = 0$, and ϕ as in Lemma 18, and when the sampling is such that either:*

- *All communication edges are sampled at once, or*
- *Each node samples exactly one virtual edge.*

PROOF. First of all, $g = 0$ is separable, and ϕ is separable with respect to the communication and computation blocks by construction.

We note b_{comm} the block of all communication edges, which is sampled with probability p_{comm}. All communication edges are sampled at the same time, so $p_i = p_{\text{comm}}$ for all $i \in b_{\text{comm}}$ and so ϕ respects option 2 for the communication block.

Let us now consider a computation block b. First of all, ϕ is separable for the virtual edges. Then, virtual blocks contain exactly one virtual edge per node, and so $b = \{(1, j_1), \cdots, (n, j_n)\}$. Let $k \neq \ell$, then

$$e_{k,j_k}^\top A^\top \Sigma A e_{\ell,j_\ell} = \mu_{k,j_k} \mu_{\ell,j_\ell} (e_k - e_{k,j_k})^\top \Sigma (e_\ell - e_{\ell,j_\ell}) = 0.$$

Therefore,

$$\begin{aligned}q_A\left(x_t + \sum_{(i,j)\in b} \delta_{ij}\right) - q_A(x_t) &= \frac{1}{2}\left(\sum_{(i,j)\in b} \delta_{ij}\right) A^\top \Sigma A \left(\sum_{(ij)\in b} \delta_{ij}\right) + \left(\sum_{(ij)\in b} A\delta_{ij}\right)^\top \Sigma A x_t \\ &= \sum_{(i,j)\in b} \left(q_A(\delta_{ij}) + \delta_{ij}^\top A^\top \Sigma A x_t\right) \\ &= \sum_{(i,j)\in b} \left(q_A(x_t + \delta_{ij}) - q_A(x_t)\right).\end{aligned}$$

Finally, f^*_{sum} is separable, and so $q_A + f^*_{\text{sum}}$ respects option 2. □

We can now prove the main theorem on the convergence rate of DVR.

THEOREM 23. *We choose* $p_{\text{comm}} = \left(1 + \gamma \frac{m+\kappa_s}{\kappa_{\text{comm}}}\right)^{-1}$ *and* $p_{ij} \propto (1 - p_{\text{comm}})(1 + L_{ij}/\sigma_i)$. *Then, for all* $\theta_0 \in \mathbb{R}^{n\times d}$ *and all* $t > 0$, *the error is such that:*

$$\frac{\eta_t}{p_{\min}} D_\phi(\lambda_\star, \lambda_t) + D(\lambda_t) - D(\lambda_\star) \leq \left(1 - \frac{\alpha\eta_t}{2}\right)^t \left[\frac{\eta_t}{p_{\min}} D_\phi(\lambda_\star, \lambda_0) + D(\lambda_0) - D(\lambda_\star)\right], \quad (4.B.4)$$

with $p_{\min} = \min(p_{\text{comm}}, \min_{ij} p_{ij})$, $\lambda_t = (x_t, y_t)$ *and* $D = -(q_A + f^*_{\text{sum}})$. *Therefore, the expected time* T_ε *required to reach precision* ε *is equal to:*

$$T_\varepsilon = O\left(\left[m + \kappa_s + \tau \frac{\kappa_{\text{comm}}}{\gamma}\right] \log \varepsilon^{-1}\right).$$

PROOF. Using Lemmas 20 and 19, we apply Theorem 22 (convergence of Bregman coordinate gradient descent), and obtain that the convergence rate is $\eta_t\alpha/2$, with $\eta_t \leq \min_{ij} p_{ij}/L^i_{\text{rel}}$ and $\eta_t \leq p_{\text{comm}}/L^{\text{comm}}_{\text{rel}}$. Therefore, for communication edges, we have that

$$\eta_t \leq \frac{p_{\text{comm}}}{L^{\text{comm}}_{\text{rel}}} = \frac{p_{\text{comm}}}{\lambda_{\max}(A^\top_{\text{comm}} \Sigma^{-1}_{\text{comm}} A_{\text{comm}})}.$$

For computation edges, we know that $p_{ij} = p_{\text{comm}}(1 + L_{ij}/\sigma_i)/(\sum_{j=1}^m (1 + L_{ij}/\sigma_i))$, and so

$$\eta_t \leq \frac{p_{ij}}{L^{ij}_{\text{rel}}} = \frac{p_{\text{comp}}}{\alpha \sum_{j=1}^m (1 + \sigma_i^{-1} L_{ij})} \leq \frac{p_{\text{comp}}}{\alpha(m + \kappa_s)},$$

with $\kappa_s \geq \sigma_i^{-1} \sum_{j=1}^m L_{ij}$ for all i.

In the end, we would like these two bounds to be equal, so we choose p_{comp} and p_{comm} such that

$$p_{\text{comp}} = p_{\text{comm}} (m + \kappa_s) \frac{\lambda^+_{\min}(A^\top_{\text{comm}} D_M^{-1} A_{\text{comm}})}{\lambda_{\max}(A^\top_{\text{comm}} \Sigma^{-1}_{\text{comm}} A_{\text{comm}})}.$$

Yet, we also know that $p_{\text{comm}} = 1 - p_{\text{comp}}$, so

$$p_{\text{comp}} = \left(1 + \frac{1}{m + \kappa_s} \frac{\lambda_{\max}(A^\top_{\text{comm}} \Sigma^{-1}_{\text{comm}} A_{\text{comm}})}{\lambda^+_{\min}(A^\top_{\text{comm}} D_M^{-1} A_{\text{comm}})}\right)^{-1}.$$

Equivalently, this corresponds to taking

$$p_{\text{comm}} = \left(1 + \gamma \frac{m + \kappa_s}{\kappa_{\text{comm}}}\right)^{-1}.$$

With this choice, one can verify that η_t verifies both $\eta_t\alpha \leq 2p_{\text{comm}}$ and $\eta_t\alpha \leq 2 \min_{ij} p_{ij}$, so the rate is:

$$1 - \frac{\eta_t\alpha}{2} = 1 - \frac{p_{\text{comp}}}{2(m + \kappa_s)}.$$

The expected execution time to reach precision ε, denoted T_ε, is equal to $T_\varepsilon = \rho^{-1}(p_{\text{comp}} + \tau p_{\text{comm}})K_\varepsilon$ with K_ε such that $C(1 - \eta_t\alpha/2)^{K_\varepsilon} < \varepsilon$ for some constant C, and so:

$$T_\varepsilon = O\left(2(m + \kappa_s) + \tau \frac{\kappa_{\text{comm}}}{\gamma}\right).$$

□

4.B.2. Primal guarantees. The goal of this section is to recover primal guarantees from dual guarantees. Although the initial setting is inspired from Lin et al. [2015b], the proof is different, and in particular does not require smoothness of the f_{ij}^* or an extra proximal step. We define for $\beta \geq 0$ the Lagrangian function:

$$\mathcal{L}(\lambda, \theta) = \sum_{i=1}^{n} \sum_{j=1}^{m} f_{ij}(\theta^{(ij)}) + \frac{\sigma_i}{2}\|\theta^{(i)}\|^2 + \frac{\beta}{2}\|\theta^{(i)} - \omega^{(i)}\|^2 - \lambda^\top A^\top \theta. \tag{4.B.5}$$

The dual problem $D(\lambda)$ is defined as

$$D(\lambda) = \min_{\theta} \mathcal{L}(\lambda, \theta).$$

Given an approximate dual solution λ_k, we can get an approximate primal solution $\theta_k = \arg\min_\theta \mathcal{L}(\lambda_k, \theta)$, which is obtained as:

$$\theta_t^{(ij)} = \arg\min_v \left(f_{ij}(v) - \mu_{ij}\lambda_t^{(ij)} v \right) \in \partial f_{ij}^*(\mu_{ij}\lambda_t^{(ij)}), \tag{4.B.6}$$

$$\theta_t^{(i)} = \frac{1}{\sigma_i + \beta}\left((A\lambda_t)^{(i)} + \beta\omega^{(i)} \right). \tag{4.B.7}$$

Note that $\theta_t^{(ij)}$ corresponds to the $z_t^{(ij)}$ from Algorithm 8. We chose to use a different notation in the main text to emphasize on the fact that these are the parameters for the virtual nodes, but $z_t^{(ij)}$ actually converge to the solution as well. Similarly, λ_t corresponds to (x_t, y_t) the concatenation of the parameters for communication and virtual edges from Section 4.2. The last difference is that the Lagrangian defined in Equation (4.B.5) actually corresponds to a Lagrangian associated to a perturbed version of Problem (4.1.1) in which $\tilde{f}_i(\theta) = f_i(\theta) + \frac{\beta}{2}\|\theta - \omega^{(i)}\|^2$. The solution to the initial problem can be retrieved by taking $\beta = 0$, but this more general formulation enables us to derive results that also holds for the inner problems solved by the Catalyst accelerated version of DVR.

LEMMA 21. *Denote* $C_0 = \frac{(\beta+\sigma_{\max}+L_{\max})}{2(\sigma_{\min}+\beta)^2}\left(\frac{p_{\min}}{\eta_t} D_\phi(\lambda^\star, \lambda_0) + (D(\lambda^\star) - D(\lambda_0))\right)$, *then*

$$\sum_{i=1}^{n} \|\theta_t^{(i)} - \theta^\star\|^2 \leq C_0(1-\rho)^t. \tag{4.B.8}$$

PROOF. Using the fact that $\theta_t^{(i)} = \frac{1}{\sigma_i+\beta}((A\lambda_t)^{(i)} + \omega_t^{(i)})$ (and similarly for $\theta^\star$), where Σ_β is the block diagonal matrix such that $(\Sigma_\beta)_{ii} = (\sigma_i + \beta)^{-1} I_d$, we obtain:

$$\begin{aligned}\sum_{i=1}^{n} \|\theta_t^{(i)} - \theta^\star\|^2 &= \sum_{i=1}^{n} \frac{1}{(\sigma_i+\beta)^2}\|(A\lambda_t)^{(i)} - (A\lambda^\star)^{(i)}\|^2 \\ &\leq \frac{1}{(\sigma_{\min}+\beta)^2}\|A\lambda_t - A\lambda^\star\|^2.\end{aligned}$$

Using the $\min((\sigma_{\max}+\beta)^{-1}, L_{ij}^{-1})$-strong convexity of $\theta \mapsto \frac{1}{2}x^\top \Sigma_\beta x + \sum_{i,j} f_{ij}^*(x^{(ij)})$, we obtain:

$$\sum_{i=1}^{n} \|\theta_t^{(i)} - \theta^\star\|^2 \leq 2\frac{(\beta + \sigma_{\max} + L_{\max})}{(\sigma_{\min}+\beta)^2}\left(D(\lambda^\star) - D(\lambda_t)\right). \tag{4.B.9}$$

Then, we add $p_{\min}\eta_t^{-1} D_\phi(\lambda^\star, \lambda_t) \geq 0$ and apply Theorem 23, which yields

$$\sum_{i=1}^{n} \|\theta_t^{(i)} - \theta^\star\|^2 \leq \frac{2(\beta + \sigma_{\max} + L_{\max})}{(\sigma_{\min}+\beta)^2}(1-\rho)^t \left(\frac{p_{\min}}{\eta_t} D_\phi(\lambda^\star, \lambda_0) + (D(\lambda^\star) - D(\lambda_0))\right).$$

□

Then, Theorem 20 is a direct consequence of Theorem 23 and Lemma 21.

4.C. Catalyst acceleration

We show in this Section how to apply Catalyst acceleration to DVR, and prove the convergence speed in this case.

4.C.1. Derivation and rates. In the main text, we derived DVR to solve regularized finite sum problems. Although not so different, the subproblem obtained with Catalyst is not in the form of Problem (4.1.1), and some adjustments need to be made. More specifically, we would like to solve problems of the form:

$$\min_{\theta} \left\{ F_t(\theta) \triangleq \sum_{i=1}^{n} \left[\frac{\sigma_i}{2}\|\theta\|^2 + \frac{\beta}{2}\|\theta - \omega_t^{(i)}\|^2 + \sum_{j=1}^{m} f_{ij}(\theta) \right] \right\}. \tag{4.C.1}$$

An easy way to adapt the algorithm is to consider the extra $(\beta/2)\|\theta - \omega_t^{(i)}\|^2$ as just another component of the sum. Yet, the point of this extra term is to make the problem easier to solve by adding strong convexity. This would not be the case if this term were is treated as just another term in the sum. Therefore, we want to include it with the quadratic term. We define:

$$h(x) = \frac{\sigma_i}{2}\|\theta\|^2 + \frac{\beta}{2}\|\theta - \omega_t^{(i)}\|^2,$$

then $h^*(x) = \frac{1}{2(\beta+\sigma)}\|x + \beta\omega_t^{(i)}\|^2 - \frac{\beta}{2}\|\omega_t^{(i)}\|^2$. Therefore, Problem (4.2.3) becomes:

$$\min_{\lambda \in \mathbb{R}^{(E+mn)d}} \frac{1}{2}\lambda^\top A^\top \Sigma_\beta A\lambda + \beta\omega_t^\top \Sigma_\beta A\lambda + \sum_{i=1}^{n}\sum_{j=1}^{m} f_{ij}^*((A\lambda)_{ij}), \tag{4.C.2}$$

with $(\Sigma_\beta)_{ii} = (\sigma_i + \beta)^{-1}$ for $i \in \{1, \dots, n\}$. The linear term does not affect the Hessians, and thus the convergence rate is the same as before, with σ replaced by $\sigma + \beta$. In terms of algorithms, we just need to modify the gradient term, and obtain Algorithm 9. The only term that changes is $\nabla q_A(x, y)$, to which an extra $\beta\Sigma_\beta\omega_t$ term is added. Therefore, the updates to θ_t and z_t remain unchanged, and only the initial expression of θ_t requires some adjustments since we now have that (as written in Equation 4.B.7):

$$\theta_{t,k}^{(i)} = \frac{1}{\sigma_i + \beta}\left((A\lambda_{t,k})^{(i)} + \beta\omega^{(i)}\right).$$

If we only consider 1 inner loop then the only thing that changes is the initial condition. If we consider several outer loops, then the we must choose the new parameter as $\theta_0^{t+1} = \theta_T^t + \Sigma_\beta(\omega_{t+1} - \omega_t)$ in order to maintain the invariant, but a remarkable fact is that the inner iterations remain the same, with the only exception that Σ is replaced by Σ_β. Note that it is possible to warm-start the $z_{t+1,0}$ as well, but this requires updating $\theta_{t,0}$ accordingly with $\nabla f_{ij}(z_{t,0}^{(ij)})$, which requires a full pass over the local dataset. We therefore choose not to do it.

However, it is not obvious that Algorithm 9 corresponds to a genuine Catalyst acceleration yet. Indeed, Catalyst acceleration requires having a feasible ε_t-approximations for the primal problem, *i.e.*, points $\theta_t \in \mathbb{R}^d$ such that $F_t(\theta_t) - \min_\theta F(\theta) \leq \varepsilon_t$. In our case, we only

Algorithm 9 Accelerated DVR(z_0)

$\alpha = 2\lambda^+_{\min}(A^\top_{\text{comm}} D_M^{-1} A_{\text{comm}})$, $\eta = \min\left(\frac{p_{\text{comm}}}{\lambda_{\max}(A^\top_{\text{comm}}\Sigma_{\beta,\text{comm}}A_{\text{comm}})}, \frac{p_{ij}}{\alpha(1+\sigma_i^{-1}L_{ij})}\right)$
$q = \frac{\sigma_{\min}}{\sigma_{\min}+\beta}$ *// Initialization*
$\omega_0^{(i)} = -\frac{1}{\sigma_i+\beta}\sum_{j=1}^m \nabla f_{ij}(z_0^{(ij)})$, $\theta_0^{(i)} = \left(1+\frac{\beta}{\sigma_i+\beta}\right)\omega_0^{(i)}$. *// z_0 is arbitrary but not θ_0.*
for $t = 0$ to $T-1$ **do** *// T outer loops*
  for $k = 0$ to $K-1$ **do** *// Inner loop runs for K iterations*
    $z_{t,k+1} = z_{t,k}$.
    Sample u_t uniformly in $[0,1]$. *// Randomly decide the kind of update*
    if $u_t \leq p_{\text{comm}}$ **then**
      $\theta_{t,k+1} = \theta_{t,k} - \frac{\eta_t}{p_{\text{comm}}}\Sigma_\beta W\theta_{t,k}$ *// Communication using W*
    else
      for $i = 1$ to n **do**
        Sample $j \in \{1,\cdots,m\}$ with probability p_{ij}.
        $z_{t,k+1}^{(ij)} = \left(1-\frac{\alpha\eta}{p_{\text{comp}}}\right)z_{t,k}^{(ij)} + \frac{\alpha\eta}{p_{\text{comp}}}\theta_{t,k}^{(i)}$ *// Computing new virtual node parameter*
        $\theta_{t,k+1}^{(i)} = \theta_{t,k}^{(i)} - \frac{1}{\sigma_i+\beta}\left(\nabla f_{ij}(z_{t,k+1}^{(ij)}) - \nabla f_{ij}(z_{t,k}^{(ij)})\right)$ *// Local update using f_{ij}*
  $\omega_{t+1} = \theta_{t,K} + \frac{1-\sqrt{q}}{1+\sqrt{q}}(\theta_{t,K} - \theta_{t-1,K})$
  $\theta_{t+1,0} = \theta_{t,K} + \frac{\beta}{\beta+\sigma_i}(\omega_{t+1} - \omega_t)$
  $z_{t+1,0} = z_{t,K}$
return θ_T

have dual guarantees and approximate feasibility. We know that the parameters converge to consensus, but they do not reach it at any time. This is a problem because it is then not possible to adequately define F_{t+1} based on the local approximations of the solutions of F_t. Yet, following the approach of Li and Lin [2020], we note that

$$\sum_{i=1}^n \|\theta - \omega_t^{(i)}\|^2 = n\|\theta - \bar{\omega}_t\|^2 + \sum_{i=1}^n \|\omega_t^{(i)}\|^2 - n\|\bar{\omega}_t\|^2,$$

where $\bar{\omega}_t = \frac{1}{n}\sum_{i=1}^n \omega_t^{(i)}$. This means that although F_t is only defined with the local variables $\omega_t^{(i)}$, *solving F_t is equivalent to solving a problem involving $\bar{\omega}_t$ only.* Besides, the Catalyst iterations are linear, meaning that performing the extrapolation step on $\bar{\theta}_t$ is equivalent to performing it on each $\theta_t^{(i)}$ individually. Therefore, although Catalyst is implemented in a fully decentralized manner (each node knowing only its own parameter), it is conceptually applied to a mean parameter $\bar{\theta}_t$ (that is never explicitly computed). In the following, we thus analyze the performances of the following algorithm:

$$\begin{aligned}\bar{\theta}_{t+1} &\approx \arg\min_\theta F(\theta) + \frac{n\beta}{2}\|\theta - \bar{\omega}_t\|^2 \\ \bar{\omega}_t &= \bar{\theta}_{t+1} + \frac{1-\sqrt{q}}{1+\sqrt{q}}(\bar{\theta}_{t+1} - \bar{\theta}_t),\end{aligned} \tag{4.C.3}$$

where we recall that $q = \sigma_{\min}/(\sigma_{\min}+\beta)$. Recall that the inner problem is approximated using DVR and the means do not need to be computed explicitly. Let $\kappa_s^\beta = \max_i 1 +$

$(\sum_{j=1}^m L_{ij})/(\beta+\sigma_i)$, and $\kappa_{\text{comm}}^\beta$ be obtained similarly to κ_{comm} but replacing Σ by Σ_β. We consider in this section that $\sigma_i = \sigma$ for all $i \in \{1, \dots, n\}$ in order to simplify exposition, but the results hold more generally. Note that α and η have slightly different expressions than in the main text since β is now involved in their definitions. We define the sequence ε_t which is such that:

$$\varepsilon_t = \frac{2}{9}\left(F(\theta_0) - F(\theta^\star)\right)(1-\rho^{\text{out}})^t \text{ with } \rho^{\text{out}} < \sqrt{q}, \text{ and } q = \frac{\sigma}{\sigma+\beta}. \tag{4.C.4}$$

We then prove the following theorem:

THEOREM 24. *Consider Algorithm 9 with* $p_{\text{comm}} = \left(1+\gamma\frac{m+\kappa_s^\beta}{\kappa_{\text{comm}}^\beta}\right)^{-1}$, $p_{ij} \propto (1-p_{\text{comm}})(1+L_{ij}/(\sigma_i+\beta))$. *If* $K = \tilde{O}\left(1/(\eta_t\alpha)\right)$ *then for all* $t \leq T$, $F_t(\bar{\theta}_t) - F_t(\theta_t^\star) \leq \varepsilon_t$ *and*

$$F(\bar{\theta}_t) - F(\theta^\star) \leq \frac{8}{(\sqrt{q}-\rho^{\text{out}})^2}(1-\rho^{\text{out}})^{t+1}(F(\bar{\theta}_0) - F(\theta^\star)). \tag{4.C.5}$$

Note that the error is on the mean parameter, and we also want $\theta_t^{(i)}$ to be close to $\bar{\theta}_t$ for all i. This is ensured by Lemma 21. Before we start the proof of Theorem 24, we show that Theorem 21 is a corollary of Theorem 24.

PROOF OF THEOREM 21. Using the same argument as in Theorem 20, we obtain that each inner loop takes time

$$T_{\text{inner}} = O\left(m + \frac{L_s+\sigma}{\beta+\sigma} + \tau\frac{L_{\text{comm}}+\beta}{\gamma(\beta+\sigma)}\right)$$

in expectation, so the total number of inner iterations is of order:

$$T_\varepsilon = \tilde{O}\left(\sum_{k=0}^{\lceil 1/\rho^{\text{out}}\rceil} T_{\text{inner}}\right) = \tilde{O}\left(\sqrt{1+\frac{\beta}{\sigma}}\left(m + \frac{L_s+\sigma}{\beta+\sigma} + \tau\frac{L_{\text{comm}}+\beta}{\gamma(\beta+\sigma)}\right)\log\frac{1}{\varepsilon}\right). \tag{4.C.6}$$

Therefore, we see that if we choose $\beta+\sigma = L_{\text{comm}}$ then, taking into account the fact that $\kappa_s \leq m\kappa_{\text{comm}}$, the algorithm takes time:

$$T_\varepsilon = \tilde{O}\left(\sqrt{\kappa_{\text{comm}}}\left(m + \frac{\tau_c}{\gamma}\right)\right).$$

Therefore, using Chebyshev acceleration allows to recover the rate of optimal batch algorithms (up to log factors). On the other hand, if we choose $\beta = L_s/m - \sigma$ then if $\beta \geq 0$ (*i.e.*, $\kappa_s \geq m$), the time to convergence is equal to:

$$T_\varepsilon = \tilde{O}\left(\sqrt{\frac{\kappa_s}{m}}\left(m + \tau\frac{m\kappa_{\text{comm}}+\kappa_s}{\gamma\kappa_s}\right)\right).$$

This can be rewritten as:

$$T_\varepsilon = \tilde{O}\left(\sqrt{m\kappa_s} + \tau\frac{\sqrt{\kappa_{\text{comm}}}}{\gamma}\sqrt{\frac{m\kappa_{\text{comm}}}{\kappa_s}}\right).$$

Therefore, we obtain the optimal $\sqrt{m\kappa_s}$ computation complexity in this case, with a slightly suboptimal communication complexity due to the $\sqrt{m\kappa_{\text{comm}}/\kappa_s}$ term. When this term is equal to 1 then $\sqrt{m\kappa_s} = m\sqrt{\kappa_b}$ and so nothing is gained from using a stochastic algorithm. Otherwise, this allows to trade-off communications for computations. □

The proof of Theorem 24 is obtained in several steps, that we emphasize below:

(1) Equivalent decentralized implementation of Catalyst.
(2) Bounding the primal suboptimality as $F_t(\bar{\theta}_t) - \min_\theta F_t(\theta) \leq (1-(\eta\alpha)/2)^k D_0^t$, with k the number of inner iterations and D_0^t a dual error. This quantifies how precisely the inner problem is solved.
(3) Evaluating the initial dual suboptimality D_0^t, which depends on θ_{t-1} (and its associated dual parameter λ_{t-1}). This quantifies how good $\bar{\theta}_{t-1}$ already is as a solution to F_t.

In the end, this allows us to use the catalyst general results with primal criterion, and with simple warm-start scheme (warm-start on the last iterate of the last outer iteration). The first point is presented at the beginnning of this section and the second one is adressed by Lemma 21. The following section deals the last point.

4.C.2. Proof of Theorem 24. We now show a bound on the initial error of an inner loop when warm-starting on the last iterate of the previous inner loop. Indeed, the convergence results for DVR depend on the initial dual error and so results from [Lin et al., 2017] cannot be used directly. Yet, it can be adapted, as we show in this section. We note $D_t(\lambda)$ the dual function at outer step t (which should not be mistaken with the Bregman divergence D_ϕ), and $\lambda_\star^t$ its minimizer. Similarly, we note $\theta_\star^t = \arg\min_\theta F_t(\theta)$, whereas $\theta^\star$ is the global minimizer of F. The following theorem ensures convergence of $\bar{\theta}_t$ to the true optimum, given that the subproblems are solved precisely enough.

THEOREM 25. *[Lin et al., 2017, Proposition 5]. If $F_k(\bar{\theta}_k) - F_k(\theta_\star^k) \leq \varepsilon_k$ for all $k \leq t$ then*

$$F(\bar{\theta}_t) - F(\theta^\star) \leq \frac{8}{(\sqrt{q}-\rho^{\text{out}})^2}(1-\rho^{\text{out}})^{t+1}(F(\bar{\theta}_0) - F(\theta^\star)). \tag{4.C.7}$$

Therefore, our goal is to prove that $F_t(\bar{\theta}_{t+1}) - F_t(\theta_\star^t) \leq \varepsilon_t$ for all t. The smoothness of F_t ensures that this is achieved if

$$\sum_{i=1}^n \|\theta_{t+1}^{(i)} - \theta_\star^t\|^2 \leq \frac{n}{L}\varepsilon_t. \tag{4.C.8}$$

Yet, using Lemma 21, we know that, since $\theta_{t+1}^{(i)}$ is obtained by applying K steps of DVR to F_t starting from λ_0^t.

$$\sum_{i=1}^n \|\theta_{t+1}^{(i)} - \theta_\star^t\|^2 \leq \frac{(\beta+\sigma_{\max}+L_{\max})}{(\sigma_{\min}+\beta)^2}(1-\rho)^K\left(\frac{p_{\min}}{\eta_t}D_\phi(\lambda_\star^t,\lambda_0^t) + D_t(\lambda_\star^t) - D_t(\lambda_0^t)\right).$$

Unfortunately, we have no control over the dual error at this point. In the remainder of this section, we prove by recursion that Equation (4.C.8) holds for all t. More specifically, we start by assuming that:

$$\frac{1}{2}\sum_{i=1}^n \|\theta_{t+1}^{(i)} - \theta_\star^t\|^2 \leq \frac{n}{L}\varepsilon_t, \tag{4.C.9}$$

$$\frac{1}{2}\sum_{i=1}^n\sum_{j=1}^m \|\theta_{t+1}^{(ij)} - \theta_\star^t\|^2 \leq C_1\varepsilon_t, \tag{4.C.10}$$

$$D_t(\lambda_\star^t) - D_t(\lambda_{t+1}) \leq C_2\varepsilon_t, \tag{4.C.11}$$

where C_1 and C_2 are such that the conditions are verified for $t = -1$, with $D_{-1} = D_0$, $\theta_\star^{-1} = \theta_\star^0$, and $\lambda_\star^{-1} = \lambda_\star^0$. Equation (4.C.9) may not hold for $t = -1$, but making it hold at time $t = 0$ would only require a slightly longer first inner iteration, meaning at most an extra log factor. Therefore we assume without loss of generality that it is the case, since the final complexities are given up to logarithmic factors. The rest of this section is devoted to showing that if K is chosen as in Theorem 24 then Equations (4.C.9), (4.C.10) and (4.C.11) hold regardless of t. The first part focuses on assessing the initial error of outer iteration $t+1$ when the conditions hold at the end of outer iteration t, and the second part on showing how these errors shrink during outer iteration $t+1$.

Warm-start error. We know that DVR converges linearly, and so the error for each subproblem decreases exponentially fast. Yet, we need to know how big the error is when solving a new problem in order to make sure that the progress from solving previous subproblems is not lost. The point of this is to avoid an extra $\log(\varepsilon^{-1})$ factor in the rate, which would come from having to solve each subproblem from a $O(1)$ precision to an ε precision using DVR. We show in this section that the initial error is actually much lower than $O(1)$ and decreases with the outer iterations. We first start by bounding the variations of ω_t across iterations, which we will need for the next proofs.

LEMMA 22 (Distance between subproblems). *It holds that*

$$\|\omega_t - \omega_{t-1}\|^2 \leq C_\omega \varepsilon_{t-1}, \text{ with } C_\omega = \frac{1080n}{1-\rho^{\text{out}}}\left(\frac{8(1-\rho^{\text{out}})}{\sigma_{\min}(\sqrt{q}-\rho^{\text{out}})^2} + \frac{4}{9L}\right).$$

PROOF. The form of the updates yields that (see Lin et al. [2017, Proposition 12] or Li and Lin [2020, Proof of Lemma 10])

$$\|\omega_t^{(i)} - \omega_{t-1}^{(i)}\| \leq 40 \max\{\|\theta_t^{(i)} - \theta^\star\|, \|\theta_{t-1}^{(i)} - \theta^\star\|, \|\theta_{t-2}^{(i)} - \theta^\star\|\}.$$

Note that here, $\theta^\star$ is the actual solution of the primal problem without the catalyst perturbation. Then, the error can be decomposed as:

$$\begin{aligned}\sum_{i=1}^n \|\theta_t^{(i)} - \theta^\star\|^2 &\leq 3\sum_{i=1}^n \left(\|\theta_t^{(i)} - \theta_\star^t\|^2 + \|\theta_\star^t - \bar{\theta}_t\|^2 + \|\bar{\theta}_t - \theta^\star\|^2\right) \\ &\leq 3n\|\bar{\theta}_t - \theta^\star\|^2 + 6\sum_{i=1}^n \|\theta_t^{(i)} - \theta_\star^t\|^2.\end{aligned}$$

Finally, the strong convexity of F leads to

$$\frac{\sigma_{\min}}{2}\|\bar{\theta}_t - \theta^\star\|^2 \leq F(\bar{\theta}_t) - F(\theta^\star) \leq \frac{8}{(\sqrt{q}-\rho^{\text{out}})^2}(1-\rho^{\text{out}})^{t+1}(F(\theta_0) - F(\theta^\star)) \tag{4.C.12}$$

where in the last inequality we use [Lin et al., 2017, Proposition 5], which holds because $F_k(\bar{\theta}_k) - F_k(\theta_\star^k) \leq \varepsilon_k$ for all $k < t$. Indeed, K is such that for all $k \leq t$, $\frac{1}{2}\sum_{i=1}^n \|\theta_k^{(i)} - \theta_\star^k\|^2 \leq \frac{n}{L}\varepsilon_k$, which yields:

$$F_k(\bar{\theta}_k) - F_k(\theta_\star^k) \leq \frac{L}{2}\|\bar{\theta}_k - \theta_\star^k\|^2 \leq \frac{L}{2n}\sum_{i=1}^n \|\theta_k^{(i)} - \theta_\star^k\|^2 \leq \varepsilon_k.$$

Therefore,

$$\sum_{i=1}^n \|\theta_t^{(i)} - \theta^\star\|^2 \leq 6n\left(1-\rho^{\text{out}}\right)^t (F(\theta_0) - F(\theta^\star))\left(\frac{8(1-\rho^{\text{out}})}{\sigma_{\min}(\sqrt{q}-\rho^{\text{out}})^2} + \frac{4}{9L}\right),$$

and a similar bound can be used for $\theta_{t-1}^{(i)}$ and $\theta_{t-2}^{(i)}$. Then, we finish proof by plugging in the expression of ε_{t-1}. □

We then use Lemma 22 to bound the initial dual error. We denote θ_k^t (and λ_k^t) the parameters at inner iteration k of outer iteration t.

LEMMA 23 (Dual error warm-start). *The warm-started dual error verifies:*

$$D_t(\lambda_\star^t) - D_t(\lambda_t) \leq C_D \varepsilon_{t-1}, \text{ with } C_D = \left(C_2 + C_\omega + 4\frac{\beta n}{L}\right). \tag{4.C.13}$$

Note that we simply warm-start the dual coordinates for an outer iteration using the last iterate from the previous one. Yet, this leads to $\theta_0^t = \theta_K^{t-1} + \beta\Sigma_\beta^{-1}(\omega_{t+1} - \omega_t)$, as in Algorithm 9.

PROOF. Equation (4.C.2) implies that $D_t(\lambda)$ can be written as:

$$D_t(\lambda) = -\sum_{i=1}^n \frac{1}{\beta + \sigma_i}\left[\frac{1}{2}(A\lambda)^{(i)} + \beta\omega_t^{(i)}\right]^\top (A\lambda)^{(i)} + R_{\text{comp}}(\lambda), \tag{4.C.14}$$

with $R_{\text{comp}}(\lambda)$ that only depends on $\lambda^{(ij)}$ and not on $\omega_t^{(i)}$ for $i \in \{1, \cdots, n\}$. Therefore,

$$\begin{aligned} &D_t(\lambda_\star^t) - D_t(\lambda_K^{t-1}) \\ &= D_{t-1}(\lambda_\star^t) - D_{t-1}(\lambda_K^{t-1}) - \beta\sum_{i=1}^n \left[(A\lambda_\star^t)^{(i)} - (A\lambda_K^{t-1})^{(i)}\right]^\top \Sigma_\beta \left[\omega_t^{(i)} - \omega_{t-1}^{(i)}\right]. \end{aligned}$$

Equation (4.B.7) writes $(A\lambda_\star^t)^{(i)} = (\beta + \sigma_i)\theta_\star^t - \beta\omega_t^{(i)}$, and so:

$$A\lambda_\star^t - A\lambda_K^{t-1} = A\lambda_\star^t - A\lambda_\star^{t-1} + A\lambda_\star^{t-1} - A\lambda_K^{t-1} = \Sigma_\beta^{-1}(\theta_\star^t - \theta_\star^{t-1}) + A\lambda_\star^{t-1} - A\lambda_K^{t-1} - \beta(\omega_t - \omega_{t-1}).$$

Then, we know from the equivalent reformulation of Equation (4.C.3) that $\theta_t^\star = \arg\min F(\theta) + \frac{\beta}{2}\|\theta - \bar{\omega}_t\|^2$, so using the 1-Lipschitzness of the proximal operator yields

$$\|\theta_\star^t - \theta_\star^{t-1}\|^2 \leq \|\bar{\omega}_t - \bar{\omega}_{t-1}\|^2 \leq \frac{1}{n}\sum_{k=1}^n \|\omega_t^{(k)} - \omega_{t-1}^{(k)}\|^2 = \frac{1}{n}\|\omega_t - \omega_{t-1}\|^2. \tag{4.C.15}$$

Similarly, $\Sigma_\beta(A\lambda_\star^{t-1} - A\lambda_K^{t-1}) = \theta_\star^{t-1} - (\theta_K^{t-1})^{(i)}$, and so:

$$\begin{aligned} \sum_{i=1}^n \left[(A\lambda_\star^t)^{(i)} - (A\lambda_K^{t-1})^{(i)}\right] \Sigma_\beta \left[\omega_t^{(i)} - \omega_{t-1}^{(i)}\right] &\leq \sum_{i=1}^n \left\|\frac{(A\lambda_\star^t)^{(i)} - (A\lambda_K^{t-1})^{(i)}}{\beta + \sigma_i}\right\| \left\|\omega_t^{(i)} - \omega_{t-1}^{(i)}\right\| \\ \sum_{i=1}^n 2\|\theta_\star^t - \theta_\star^{t-1}\|^2 + 2\|\theta_\star^{t-1} - (\theta_K^{t-1})^{(i)}\|^2 &+ \left(\frac{\beta}{\beta + \sigma_i} + 4\right)\|\omega_t^{(i)} - \omega_{t-1}^{(i)}\|^2. \end{aligned}$$

Plugging in Equation (4.C.15) yields:

$$D_t(\lambda_\star^t) - D_t(\lambda_K^{t-1}) \leq D_{t-1}(\lambda_\star^t) - D_{t-1}(\lambda_K^{t-1}) + 2\beta\sum_{i=1}^n \|(\theta_K^{t-1})^{(i)} - \theta_\star^{t-1}\|^2 + 7\beta\|\omega_t - \omega_{t-1}\|^2$$

Finally note that $D_{t-1}(\lambda_\star^t) \leq D_{t-1}(\lambda_\star^{t-1})$ since $\lambda_\star^{t-1}$ is the maximizer of D_{t-1}, and $(\theta_K^{t-1})^{(i)} = \theta_t^{(i)}$ since it is the output of DVR after inner iteration t. The final expression is obtained using 22 and the recursion assumptions given by Equations (4.C.9) and (4.C.11). □

Finally, the warm-start error on the nodes parameters is given by the two following lemmas.

LEMMA 24 (Virtual parameters warm-starts). *Denote* $\|\theta_1 - \theta_2\|^2_{\text{comp}} = \sum_{i=1}^n \sum_{j=1}^m \|\theta_1^{(ij)} - \theta_2^{(ij)}\|^2$. *Then,*

$$\|\theta_0^t - \theta_\star^t\|^2_{\text{comp}} \leq 2(C_\omega + 2mC_1)\varepsilon_{t-1}. \tag{4.C.16}$$

PROOF. We use the fact that $(\theta_t)^{(ij)} = (\theta_0^t)^{(ij)} = (\theta_K^{t-1})^{(ij)}$ to write:

$$\|\theta_0^t - \theta_\star^t\|^2_{\text{comp}} = \|\theta_K^{t-1} - \theta_\star^{t-1} + \theta_\star^{t-1} - \theta_\star^t\|^2_{\text{comp}} \leq 2\|\theta_t - \theta_\star^{t-1}\|^2_{\text{comp}} + 2nm\|\theta_\star^{t-1} - \theta_\star^t\|^2.$$

Then, as before, the 1-Lipchitzness of the prox operator yields $\|\theta_\star^{t-1} - \theta_\star^t\| \leq \frac{1}{n}\|\omega_t - \omega_{t-1}\|$. □

LEMMA 25 (Parameters warm-start). *Denote* $\|\theta_1 - \theta_2\|^2_{\text{comp}} = \sum_{i=1}^n \sum_{j=1}^m \|\theta_1^{(ij)} - \theta_2^{(ij)}\|^2$. *Then,*

$$\sum_{i=1}^n \|(\theta_0^t)^{(i)} - \theta_\star^t\|^2 \leq 6\left(C_\omega + \frac{n}{L}\right)\varepsilon_{t-1}. \tag{4.C.17}$$

PROOF. We use the fact that since $\lambda_0^t = \lambda_K^{t-1}$ then $(\theta_0^t)^{(i)} = (\theta_0^t)^{(i)} + \frac{\beta}{\beta+\sigma_i}(\omega_t^{(i)} - \omega_{t-1}^{(i)})$ to write:

$$\begin{aligned}\sum_{i=1}^n \|(\theta_0^t)^{(i)} - \theta_\star^t\|^2 &\leq \sum_{i=1}^n \|(\theta_K^{t-1})^{(i)} - \theta_\star^{t-1} + \theta_\star^{t-1} - \theta_\star^t + \frac{\beta}{\sigma_i + \beta}(\omega_t^{(i)} - \omega_{t-1}^{(i)})\|^2 \\ &\leq 3\|\omega_t - \omega_{t-1}\|^2 + 3n\|\theta_\star^{t-1} - \theta_\star^t\|^2 + 3\sum_{i=1}^n \|(\theta_t)^{(i)} - \theta_\star^{t-1}\|^2.\end{aligned}$$

□

We finish this part on warm starts by proving the following lemma, that links the initial dual parameters error (computed with the Bregman divergence of ϕ), to the other parameters which we already know how to control.

LEMMA 26 (Dual parameters warm-start, as measured by the Bregman divergence).

$$D_\phi(\lambda_\star^t, \lambda_0^t) \leq C_\phi \varepsilon_{t-1}, \tag{4.C.18}$$

with $C_\phi = \frac{6(C_\omega + n/L) + L^2_{\max}(2C_\omega + 2mC_1)}{\lambda^+_{\min}(A^\top \Sigma_\beta^2 A)} + \frac{2L_{\max}(C_\omega + 2mC_1)}{\alpha}$.

PROOF. We first decompose the Bregman divergence as:

$$D_\phi(\lambda_0^t, \lambda_\star^t) \leq \frac{1}{2}\|(\lambda_0^t)^{\text{comm}} - (\lambda_\star^t)^{\text{comm}}\|_{A^\dagger_{\text{comm}} A_{\text{comm}}} + \sum_{i=1}^n \sum_{j=1}^m D_{\phi_{ij}}((\lambda_\star^t)^{(ij)}, (\lambda_0^t)^{(ij)}). \tag{4.C.19}$$

Then, we bound the communication term as:

$$\begin{aligned}\|(\lambda_0^t)^{\text{comm}} - (\lambda_\star^t)^{\text{comm}}\|_{A^\dagger_{\text{comm}} A_{\text{comm}}} &\leq \|\lambda_0^t - \lambda_\star^t\|_{A^\dagger A} \leq \frac{1}{\lambda^+_{\min}(A^\top \Sigma_\beta^2 A)}\|\Sigma_\beta A\left(\lambda_0^t - \lambda_\star^t\right)\|^2 \\ &= \frac{1}{\lambda^+_{\min}(A^\top \Sigma_\beta^2 A)}\left(\sum_{i=1}^n \|(\theta_0^t)^{(i)} - \theta_\star^t\|^2 + \sum_{i=1}^n \sum_{j=1}^m \mu_{ij}^2 \|(\lambda_0^t)^{(ij)} - (\lambda_\star^t)^{(ij)}\|^2\right) \\ &= \frac{1}{\lambda^+_{\min}(A^\top \Sigma_\beta^2 A)}\left(\sum_{i=1}^n \|(\theta_0^t)^{(i)} - \theta_\star^t\|^2 + \sum_{i=1}^n \sum_{j=1}^m \|\nabla f_{ij}((\theta_0^t)^{(ij)}) - \nabla f_{ij}((\theta_\star^t)^{(ij)})\|^2\right)\end{aligned}$$

$$= \frac{1}{\lambda^+_{\min}(A^\top \Sigma_\beta^2 A)} \left(\sum_{i=1}^n \|(\theta_0^t)^{(i)} - \theta_\star^t\|^2 + \sum_{i=1}^n \sum_{j=1}^m L_{ij}^2 \|(\theta_0^t)^{(ij)} - (\theta_\star^t)^{(ij)}\|^2 \right).$$

Using Lemmas 24 and 25, we obtain:

$$\frac{1}{2}\|(\lambda_0^t)^{\text{comm}} - (\lambda_\star^t)^{\text{comm}}\|_{A^\dagger_{\text{comm}} A_{\text{comm}}} \leq \frac{\left(6\left(C_\omega + \frac{n}{L}\right) + L_{\max}^2(2C_\omega + 2mC_1)\right)}{\lambda^+_{\min}(A^\top \Sigma_\beta^2 A)} \varepsilon_{t-1}. \tag{4.C.20}$$

For the computation part, we use the duality property of the Bregman divergence, which yields

$$\begin{aligned} D_{\phi_{ij}}((\lambda_\star^t)^{(ij)}, (\lambda_0^t)^{(ij)}) &= \frac{L_{ij}}{\mu_{ij}^2} D_{f_{ij}^*}(\mu_{ij}(\lambda_\star^t)^{(ij)}, \mu_{ij}(\lambda_0^t)^{(ij)}) \\ &= \frac{L_{ij}}{\mu_{ij}^2} D_{f_{ij}}(\nabla f_{ij}(\mu_{ij}(\lambda_0^t)^{(ij)}), \nabla f_{ij}(\mu_{ij}(\lambda_\star^t)^{(ij)})) \\ &= \frac{L_{ij}}{\mu_{ij}^2} D_{f_{ij}}((\theta_0^t)^{(ij)}, (\theta_\star^t)^{(ij)}) \leq \frac{L_{ij}^2}{\mu_{ij}^2} \|(\theta_0^t)^{(ij)} - (\theta_\star^t)^{(ij)}\|^2 \end{aligned}$$

Therefore,

$$\sum_{i=1}^n \sum_{j=1}^m D_{\phi_{ij}}((\lambda_0^t)^{(ij)}, (\lambda_\star^t)^{(ij)}) \leq \frac{2L_{\max}(C_\omega + 2mC_1)}{\alpha} \varepsilon_{t-1}. \tag{4.C.21}$$

Substituting Equations (4.C.20) and (4.C.21) into Equation (4.C.19) finishes the proof. □

Inner iteration error decrease. Now that we have bounded the error at the beginning of each outer iteration, we bound error at the end of each outer iteration by using the convergence results for DVR. We first prove the following Lemma, which controls the distance between the virtual parameters and the actual one:

LEMMA 27 (Virtual error decrease). *For all* (i, j),

$$\mathbb{E}[\sum_{i,j} \|(\theta_{t+1})^{(ij)} - \theta_\star^t\|^2] \leq (1-\rho)^K \left[\|\theta_0^t - \theta_\star^t\|_{\text{comp}}^2 + \frac{\rho_{\text{sum}} K}{1-\rho} C_0(t) \right]. \tag{4.C.22}$$

PROOF. We cannot retrieve direct control over the $\theta_{t+1}^{(ij)}$ from control over the dual variables or the dual error, since this would require the f_{ij}^* functions to be smooth, which they may not be. Yet, we leverage the fact that $\theta_{t+1}^{(ij)}$ is obtained by a convex combination between $\theta_t^{(ij)}$ and $\theta_t^{(i)}$ to obtain convergence of to $\theta_\star^t$. We note $j_{t,k}(i)$ the virtual node that is updated at time (t, k) for node i. We note $\mathbb{E}_k$ the expectation relative to the value of $j_{t,k}(i)$. We start by remarking that:

$$\begin{aligned} &\mathbb{E}_{k+1}\left[\|(\theta_{k+1}^t)^{(ij)} - \theta_\star^t\|^2\right] \\ &= (1-p_{ij})\|(\theta_k^t)^{(ij)} - \theta_\star^t\|^2 + p_{ij}\|(1-\rho_{ij})(\theta_k^t)^{(ij)} + \rho_{ij}(\theta_k^t)^{(i)} - \theta_\star^t\|^2 \\ &\leq (1 - p_{ij}\rho_{ij})\|(\theta_k^t)^{(ij)} - \theta_\star^t\|^2 + p_{ij}\rho_{ij}\|(\theta_k^t)^{(i)} - \theta_\star^t\|^2, \end{aligned}$$

where in the last inequality we used the convexity of the squared norm. We use that $p_{ij}\rho_{ij} \geq \rho$ (equal for the smallest one), and write that:

$$\mathbb{E}[\|(\theta_K^t)^{(ij)} - \theta_\star^t\|^2] \leq (1-\rho)^K \|(\theta_0^t)^{(ij)} - \theta_\star^t\|^2 + p_{ij}\rho_{ij} \sum_{k=1}^{K} (1-\rho)^{k-1} \|(\theta_{K-k}^t)^{(i)} - \theta_\star^t\|^2. \quad (4.C.23)$$

Noting $\rho_{\text{sum}} = \max_i \sum_{j=1}^m \rho_{ij} p_{ij}$ and $\|\theta_k^t - \theta_\star^t\|_{\text{comp},i}^2 = \sum_{j=1}^m \|(\theta_k^t)^{(ij)} - \theta_\star^t\|^2$, we obtain

$$\mathbb{E}[\|\theta_K^t - \theta_\star^t\|_{\text{comp},i}^2] \leq (1-\rho)^K \|\theta_0^t - \theta_\star^t\|_{\text{comp},i}^2 + \rho_{\text{sum}} \sum_{k=1}^{K} (1-\rho)^{k-1} \|(\theta_{K-k}^t)^{(i)} - \theta_\star^t\|^2. \quad (4.C.24)$$

Using Lemma 21, we know that $\sum_{i=1}^n \|(\theta_k^t)^{(i)} - \theta_\star^t\|^2 \leq C_0(t)(1-\rho)^k$, with $C_0(t)$ a constant that depends on the initial conditions of outer iteration t. Therefore,

$$\sum_{i=1}^{n} \sum_{k=1}^{K} (1-\rho)^{k-1} \|(\theta_{K-k}^t)^{(i)} - \theta_\star^t\|^2 \leq K(1-\rho)^{K-1} C_0(t). \quad (4.C.25)$$

In the end,

$$\mathbb{E}[\|\theta_K^t - \theta_\star^t\|_{\text{comp}}^2] \leq (1-\rho)^K \left[\|\theta_0^t - \theta_\star^t\|_{\text{comp}}^2 + \frac{\rho_{\text{sum}} K}{1-\rho} C_0(t) \right]. \quad (4.C.26)$$

□

This lemma has the following corollary:

Corollary 9 (Warm-started virtual error decrease). *For all (i,j),*

$$\mathbb{E}[\sum_{i,j} \|(\theta_{t+1})^{(ij)} - \theta_\star^t\|^2] \leq (1-\rho)^K \left[6\left(C_\omega + \frac{n}{L}\right) + K\frac{\rho_{\text{sum}} C_{\text{comp}}}{1-\rho} \right] \varepsilon_{t-1}, \quad (4.C.27)$$

with

$$C_{\text{comp}} = \frac{(\beta + \sigma_{\max} + L_{\max})}{(\sigma_{\min} + \beta)^2} \left(\frac{p_{\min}}{\eta_t} C_\phi + C_2 + C_\omega + 4\frac{\beta n}{L} \right)$$

Proof. Using Lemmas 21, 26 and 23, we write:

$$C_0(t) = \frac{(\beta + \sigma_{\max} + L_{\max})}{(\sigma_{\min} + \beta)^2} \left(\frac{p_{\min}}{\eta_t} D_\phi(\lambda_\star^t, \lambda_0^t) + \left(D(\lambda_\star^t) - D(\lambda_0^t) \right) \right) \leq C_{\text{comp}} \varepsilon_{t-1}$$

We use Lemma 24 for the first term. □

Lemma 28 (Condition on K). *If Equations (4.C.9), (4.C.10) and (4.C.11) hold at time t, and K is such that:*

$$(1-\rho)^K \leq \min \left(\frac{C_1(1-\rho^{\text{out}})}{12(C_\omega + n/L)}, \frac{C_1(1-\rho^{\text{out}})(1-\rho)}{K\rho_{\text{sum}} C_{\text{comp}}}, \frac{C_2}{C_L}, \frac{n(\sigma_{\min} + \beta)^2}{2LC_L(\beta + \sigma_{\max} + L_{\max})} \right),$$

then they also hold at time $t+1$.

Proof. Using Corollary 9, we obtain that if K is set such that

$$(1-\rho)^K \left[6\left(C_\omega + \frac{n}{L}\right) + K\frac{\rho_{\text{sum}} C_{\text{comp}}}{1-\rho} \right] \leq C_1(1-\rho^{\text{out}}),$$

then the recursion condition is respected for the virtual parameters. This yields the first and second conditions on K. Now, we write $C_L = \left(\frac{p_{\min}}{\eta_t} C_\phi + C_D\right)$, then using Lemmas 26 and 23 (where C_ϕ and C_D are defined), we obtain using Theorem 23 that

$$D_t(\lambda_\star^t) - D_t(\lambda_{t+1}) \leq C_L(1-\rho)^K \varepsilon_{t-1},$$

since λ_{t+1} is obtained by performing K iterations of DVR to minimize F_t starting from λ_t. This yields the third condition on K. Finally, the last condition on K is obtained by leveraging Lemma 21.

□

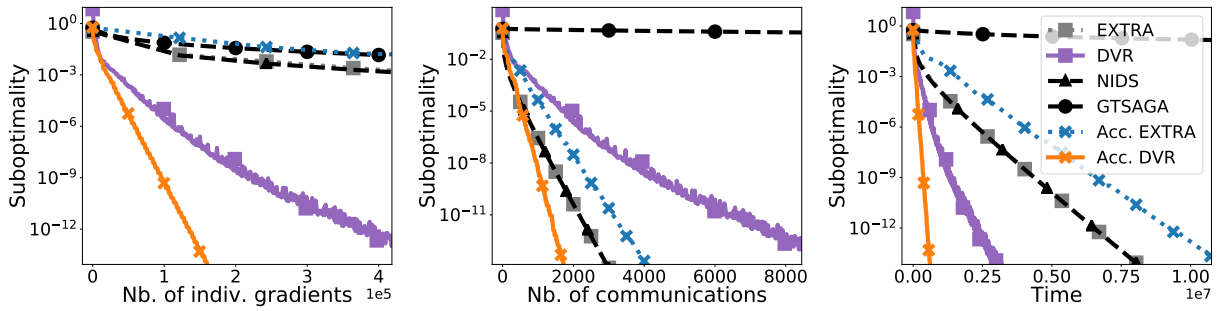
4.D. Experiments

For the experiments, the following logistic regression problem is solved:

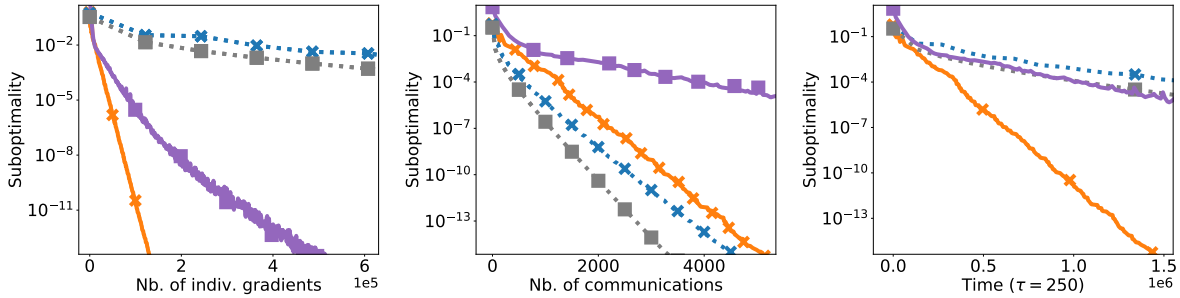
$$\min_{\theta \in \mathbb{R}^d} \sum_{i=1}^n \left[\frac{\sigma}{2} \|\theta\|^2 + \sum_{j=1}^m \frac{1}{m} \log(1 + \exp(-y_{ij} X_{ij}^\top \theta)) \right], \tag{4.D.1}$$

where the pairs $(X_{ij}, y_{ij}) \in \mathbb{R}^d \times \{-1, 1\}$ are taken from the RCV1 dataset, which we downloaded from https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/binary.html.

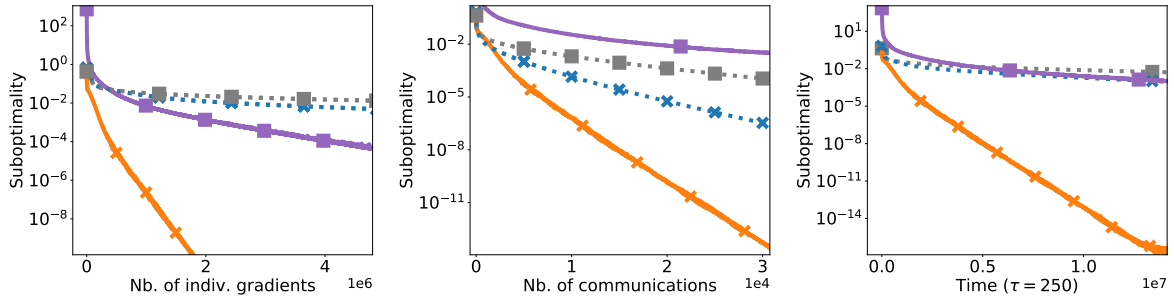
Figure 3 is the full version of Figure 1, in which we report the number of individual gradients and number of communications for each configuration. We see that accelerated EXTRA actually outperforms EXTRA when the regularization is small, as already mentioned in the main text. We also see that Accelerated EXTRA and Accelerated DVR have comparable communication complexity on the grid graph, when γ is smaller. Yet, the computation complexity of (accelerated) DVR is much smaller, so accelerated DVR is much faster overall as long as τ is not too big.



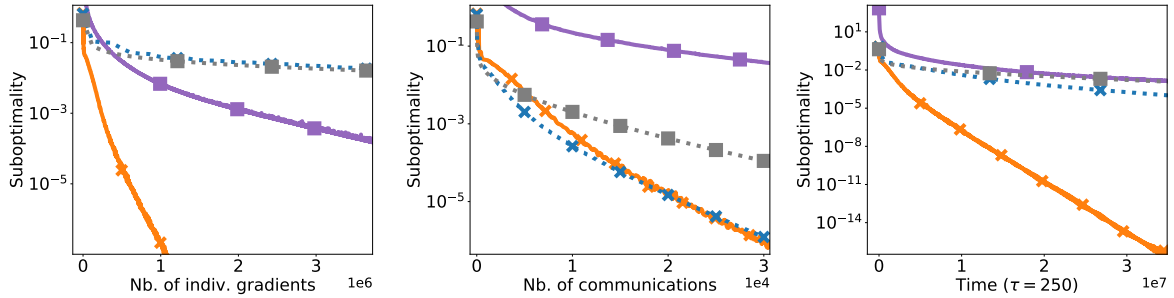
(a) Erdős-Rényi, $\sigma = m \cdot 10^{-5}$



(b) Grid, $\sigma = m \cdot 10^{-5}$



(c) Erdős-Rényi, $\sigma = m \cdot 10^{-7}$



(d) Grid, $\sigma = m \cdot 10^{-7}$

FIGURE 3. Experimental results for the RCV1 dataset with different graphs of size $n = 81$, with $m = 2430$ samples per node, and with different regularization parameters.

CHAPTER 5

Statistically Preconditioned Accelerated Gradient Method

In this Chapter, we consider the setting of distributed empirical risk minimization where multiple machines compute the gradients in parallel and a centralized server updates the model parameters. In order to reduce the number of communications required to reach a given accuracy, we propose a *preconditioned* accelerated gradient method where the preconditioning is done by solving a local optimization problem over a subsampled dataset at the server. The convergence rate of the method depends on the square root of the relative condition number between the global and local loss functions. We estimate the relative condition number for linear prediction models by studying *uniform* concentration of the Hessians over a bounded domain, which allows us to derive improved convergence rates for existing preconditioned gradient methods and our accelerated method. Experiments on real-world datasets illustrate the benefits of acceleration in the ill-conditioned regime.

This Chapter is based on the paper *Statistically Preconditioned Accelerated Gradient Method for Distributed Optimization* [Hendrikx, Xiao, Bubeck, Bach, and Massoulie, 2020c], published at ICML 2020. Other Bregman methods can be used for statistical preconditioning, and we present in particular stochastic methods from the paper *Fast Stochastic Bregman Gradient Methods: Sharp Analysis and Variance Reduction* [Dragomir, Even, and Hendrikx, 2021a] in Section 5.7.

5.1. Introduction

We consider empirical risk minimization problems of the form

$$\underset{x\in\mathbb{R}^d}{\text{minimize}}\ \Phi(x) \triangleq F(x) + \psi(x), \tag{5.1.1}$$

where F is the empirical risk over a dataset $\{z_1, \dots, z_N\}$:

$$F(x) = \frac{1}{N}\sum_{i=1}^{N} \ell(x, z_i), \tag{5.1.2}$$

and ψ is a convex regularization function. We incorporate smooth regularizations such as squared Euclidean norms $(\lambda/2)\|x\|^2$ into the individual loss functions $\ell(x, z_i)$, and leave ψ mainly for non-smooth regularizations such as the ℓ_1-norm or the indicator function of a constraint set.

In modern machine learning applications, the dataset is often very large and has to be stored at multiple machines. For simplicity of presentation, we assume $N = mn$, where m is the number of machines and n is the number of samples stored at each machine. Let $\mathcal{D}_j = \{z_1^{(j)}, \dots, z_n^{(j)}\}$ denote the dataset at machine j and define the local empirical risk

$$f_j(x) = \frac{1}{n}\sum_{i=1}^{n} \ell\Big(x, z_i^{(j)}\Big), \quad j = 1, \dots, m. \tag{5.1.3}$$

The overall empirical risk of Equation (5.1.2) can then be written as

$$F(x) = \frac{1}{m}\sum_{j=1}^{m} f_j(x) = \frac{1}{nm}\sum_{j=1}^{m}\sum_{i=1}^{n} \ell\Big(x, z_i^{(j)}\Big).$$

We assume that F is L_F-smooth and σ_F-strongly convex over $\mathrm{dom}\psi$, in other words,

$$\sigma_F I_d \preceq \nabla^2 F(x) \preceq L_F I_d, \quad \forall x \in \mathrm{dom}\psi, \tag{5.1.4}$$

where I_d is the $d \times d$ identity matrix. The condition number of F is defined as $\kappa_F = L_F/\sigma_F$.

We focus on a basic setting of distributed optimization where the m machines (workers) compute the gradients in parallel and a centralized server updates the variable x. Specifically, during each iteration $t = 0, 1, 2, \ldots$,

(i) the server broadcasts x_t to all m machines;
(ii) each machine j computes the gradient $\nabla f_j(x_t)$ and sends it back to the server;
(iii) the server forms $\nabla F(x_t) = \frac{1}{m}\sum_{j=1}^{m} \nabla f_j(x_t)$ and uses it to compute the next iterate x_{t+1}.

A standard way for solving problem (5.1.1) in this setting is to implement the proximal gradient method at the server:

$$x_{t+1} = \operatorname*{argmin}_{x\in\mathbb{R}^d}\Big\{\nabla F(x_t)^\top x + \psi(x) + \frac{1}{2\eta_t}\|x - x_t\|^2\Big\}, \tag{5.1.5}$$

where $\|\cdot\|$ denotes the Euclidean norm and $\eta_t > 0$ is the step size. Setting $\eta_t = 1/L_F$ leads to linear convergence:

$$\Phi(x_t) - \Phi(x_*) \le \Big(1 - \kappa_F^{-1}\Big)^t \tfrac{L_F}{2}\|x_* - x_0\|^2, \tag{5.1.6}$$

where $x_* = \arg\min \Phi(x)$ [e.g., Beck, 2017, Section 10.6]. In other words, in order to reach $\Phi(x_t) - \Phi(x_*) \le \epsilon$, we need $O(\kappa_F \log(1/\epsilon))$ iterations, which is also the number of communication rounds between the workers and the server. If we use accelerated proximal gradient methods [e.g., Nesterov, 2004, Beck and Teboulle, 2009a, Nesterov, 2013b] at the server, then the iteration/communication complexity can be improved to $O(\sqrt{\kappa_F}\log(1/\epsilon))$.

5.1.1. Statistical Preconditioning. In general, for minimizing $F(x) = (1/m)\sum_{j=1}^{m} f_j(x)$ with first-order methods, the communication complexity of $O(\sqrt{\kappa_F}\log(1/\epsilon))$ cannot be improved [Arjevani and Shamir, 2015, Scaman et al., 2017a]. However, for distributed empirical risk minimization (ERM), the additional finite-sum structure of each f_j in (5.1.3) allows further improvement. A key insight here is that if the datasets $\mathcal{D}_j$ at different workers are i.i.d. samples from the same source distribution, then the local empirical losses f_j are statistically very similar to each other and to their average F, especially when n is large. *Statistical preconditioning* is a technique to further reduce communication complexity based on this insight.

An essential tool for preconditioning in first-order methods is the Bregman divergence. The Bregman divergence of a strictly convex and differentiable function ϕ is defined as

$$D_\phi(x, y) \triangleq \phi(x) - \phi(y) - \nabla\phi(y)^\top(x - y). \tag{5.1.7}$$

We also need the following concepts of relative smoothness and strong convexity Bauschke et al. [2017], Lu et al. [2018].

DEFINITION 5. *Suppose $\phi : \mathbb{R}^d \to \mathbb{R}$ is convex and twice differentiable. The function F is said to be $L_{F/\phi}$-smooth and $\sigma_{F/\phi}$-strongly convex with respect to ϕ if for all $x \in \mathbb{R}^d$,*

$$\sigma_{F/\phi}\nabla^2\phi(x) \preceq \nabla^2 F(x) \preceq L_{F/\phi}\nabla^2\phi(x). \tag{5.1.8}$$

The classical definition in (5.1.4) can be viewed as relative smoothness and strong convexity where $\phi(x) = (1/2)\|x\|^2$. Moreover, it can be shown that (5.1.8) holds if and only if for all $x, y \in \mathbb{R}^d$

$$\sigma_{F/\phi}D_\phi(x, y) \le D_F(x, y) \le L_{F/\phi}D_\phi(x, y). \tag{5.1.9}$$

Consequently, we define the relative condition number of F with respect to ϕ as $\kappa_{F/\phi} = L_{F/\phi}/\sigma_{F/\phi}$.

Following the Distributed Approximate Newton (DANE) method by Shamir et al. [2014], we construct the reference function ϕ by adding some extra regularization to one of the local loss functions (say f_1, without loss of generality):

$$\phi(x) = f_1(x) + \frac{\mu}{2}\|x\|^2. \tag{5.1.10}$$

Then we replace $(1/2)\|x - x_t\|^2$ in the proximal gradient method (5.1.5) with the Bregman divergence of ϕ, i.e.,

$$x_{t+1} = \operatorname*{argmin}_{x\in\mathbb{R}^d}\left\{\nabla F(x_t)^\top x + \psi(x) + \frac{1}{\eta_t}D_\phi(x, x_t)\right\}. \tag{5.1.11}$$

In this case, worker 1 acts as the server to compute x_{t+1}, which requires solving a nontrivial optimization problem involving the local loss function f_1.

According to Shamir et al. [2014] and Lu et al. [2018], with $\eta_t = 1/L_{F/\phi}$, the sequence $\{x_t\}$ generated by (5.1.11) satisfies

$$\Phi(x_t) - \Phi(x_*) \le \left(1 - \kappa_{F/\phi}^{-1}\right)^t L_{F/\phi}D_\phi(x_*, x_0), \tag{5.1.12}$$

which is a direct extension of (5.1.6). Therefore, the effectiveness of preconditioning hinges on how much smaller $\kappa_{F/\phi}$ is compared to κ_F. Roughly speaking, the better f_1 or ϕ approximates F, the smaller $\kappa_{F/\phi}$ (≥ 1) is. In the extreme case of $f_1 \equiv F$ (with only one machine $m = 1$), we can choose $\mu = 0$ and thus $\phi \equiv F$, which leads to $\kappa_{F/\phi} = 1$, and we obtain the solution within one step.

In general, we choose μ to be an upper bound on the spectral norm of the matrix difference $\nabla^2 f_1 - \nabla^2 F$. Specifically, we assume that *with high probability*, for the operator norm between matrices (i.e., the largest singular value),

$$\left\|\nabla^2 f_1(x) - \nabla^2 F(x)\right\| \le \mu, \quad \forall\, x \in \mathrm{dom}\psi, \tag{5.1.13}$$

which implies [Zhang and Xiao, 2018, Lemma 3],

$$\frac{\sigma_F}{\sigma_F + 2\mu}\nabla^2\phi(x) \preceq \nabla^2 F(x) \preceq \nabla^2\phi(x). \tag{5.1.14}$$

Now we invoke a statistical argument based on the empirical average structure in (5.1.3). Without loss of generality, we assume that $\mathcal{D}_1$ contains the first n samples of $\{z_1, \ldots, z_N\}$

and thus $\nabla^2 f_1(x) = \frac{1}{n}\sum_{i=1}^n \nabla^2 \ell(x, z_i)$. For any fixed x, we can use Hoeffding's inequality for matrices [Tropp, 2015] to obtain, with probability $1-\delta$,

$$\left\| \frac{1}{n}\sum_{i=1}^n \nabla^2 \ell(x, z_i) - \nabla^2 F(x) \right\| \leq \sqrt{\frac{32 L_\ell^2 \log(d/\delta)}{n}}, \tag{5.1.15}$$

where L_ℓ is the uniform upper bound on $\|\nabla^2 \ell(x, z_i)\|$.

If the losses $\ell(x, z_i)$ are *quadratic* in x, then the Hessians are constant and (5.1.13) holds with $\mu = \widetilde{O}(L_\ell/\sqrt{n})$, hiding the factor $\log(d/\delta)$. In this case, we derive from (5.1.14) that

$$\kappa_{F/\phi} = 1 + \frac{2\mu}{\sigma_F} = 1 + \widetilde{O}\left(\frac{\kappa_\ell}{\sqrt{n}}\right), \tag{5.1.16}$$

where we assume $\sigma_F \approx \sigma_\ell$, where $\nabla^2 \ell(x, z_i) \succeq \sigma_\ell I_d$ for all x. Therefore, for large n, whenever we have $\kappa_{F/\phi} < \kappa_F$, the communication complexity $O(\kappa_{F/\phi} \log(1/\epsilon))$ is better than without preconditioning.

For non-quadratic loss functions, we need to ensure that (5.1.13) holds *uniformly* over a compact domain with high probability. Standard ball-packing arguments encounter an additional factor of $\sqrt{d}$ [e.g., Zhang and Xiao, 2018, Lemma 6]. In this case, we have $\mu = \widetilde{O}(L_\ell\sqrt{d/n})$ and

$$\kappa_{F/\phi} = 1 + \frac{2\mu}{\sigma_F} = 1 + \widetilde{O}\left(\frac{\kappa_\ell\sqrt{d}}{\sqrt{n}}\right), \tag{5.1.17}$$

which suggests that the benefit of preconditioning may degrade or disappear in high dimension.

5.1.2. Contributions and Outline. In this paper, we make the following two contributions.

First, we propose a Statistically Preconditioned Accelerated Gradient (SPAG) method that can further reduce the communication complexity. Accelerated methods with a complexity of $O(\sqrt{\kappa_{F/\phi}}\log(1/\epsilon))$ have been developed for quadratic loss functions (see related works in Section 5.2). However, Dragomir et al. [2019] have shown that acceleration is not possible in general in the relatively smooth and strongly convex setting, and that more assumptions are needed. Here, by leveraging the fact the reference function ϕ itself is smooth and strongly convex, we obtain

$$\Phi(x_t) - \Phi(x_*) \leq \prod_{\tau=1}^t \left(1 - \frac{1}{\sqrt{\kappa_{F/\phi} G_\tau}}\right) L_{F/\phi} D_\phi(x_*, x_0),$$

where $1 \leq G_t \leq \kappa_\phi$ and $G_t \to 1$ geometrically. Moreover, G_t can be calculated at each iteration and serve as numerical certificate of the actual convergence rate. In all of our experiments, we observe $G_t \approx 1$ even in early iterations, which results in $O(\sqrt{\kappa_{F/\phi}}\log(1/\epsilon))$ iterations empirically.

Second, we derive refined bounds on the relative condition number for linear prediction models. Linear models such as logistic regression have the form $\ell(x, z_i) = \ell_i(a_i^\top x) + (\lambda/2)\|x\|^2$. Assume that $\ell_i''(a_i^\top x) \leq 1$ and $\|a_i\| \leq R$ for all i, which implies $L_\ell = R^2$ and $\kappa_\ell = R^2/\lambda$. Then the Hoeffding bounds in (5.1.16) for quadratics becomes $\kappa_{F/\phi} = 1 + \widetilde{O}\left(\frac{R^2}{\sqrt{n}\lambda}\right)$, and

for nonquadratics, the bound in (5.1.17) (from previous work) becomes $\kappa_{F/\phi} = 1+\widetilde{O}\left(\frac{R^2\sqrt{d}}{\sqrt{n}\lambda}\right)$. We show that:

- For quadratic losses, the bound on relative condition number can be improved by a factor of $\sqrt{n}$, i.e.,
$$\kappa_{F/\phi} = \frac{3}{2} + O\left(\frac{R^2}{n\lambda}\log\left(\frac{d}{\delta}\right)\right).$$
- For non-quadratic losses, we derive a uniform concentration bound to remove the dependence of $\kappa_{F/\phi}$ on d,
$$\kappa_{F/\phi} = 1 + O\left(\frac{R^2}{\sqrt{n}\lambda}\left(RD + \sqrt{\log(1/\delta)}\right)\right),$$
where D is the diameter of $\mathrm{dom}\phi$ (bounded domain). We also give a refined bound when the inputs a_i are sub-Gaussian.

These new bounds on $\kappa_{F/\phi}$ improve the convergence rates for all existing accelerated and non-accelerated preconditioned gradient methods (see related work in Section 5.2).

We start by discussing related work in Section 5.2. In Section 5.3, we introduce SPAG and give its convergence analysis. In Section 5.4, we derive sharp bounds on the relative condition number, and discuss their implications on the convergence rates of SPAG and other preconditioned gradient methods. We present experimental results in Section 5.5.

5.2. Related Work

Shamir et al. [2014] considered the case $\psi \equiv 0$ and introduced the statistical preconditioner (5.1.10) in DANE. Yet, they define a separate $\phi_j(x) = f_j(x) + (\mu/2)\|x\|^2$ for each worker j, compute m separate local updates using (5.1.11), and then use their average as x_{t+1}. For quadratic losses, they obtain the communication complexity $\widetilde{O}((\kappa_\ell^2/n)\log(1/\epsilon))$, which is roughly $O(\kappa_{F/\phi}^2\log(1/\epsilon))$ in our notation, which is much worse than their result without averaging of $O(\kappa_{F/\phi}\log(1/\epsilon))$ given in Section 5.1.1. We further improve this to $O(\sqrt{\kappa_{F/\phi}}\log(1/\epsilon))$ using acceleration.

Zhang and Xiao [2015] proposed DiSCO, an inexact damped Newton method, where the Newton steps are computed by a distributed conjugate gradient method with a similar preconditioner as (5.1.10). They obtain a communication complexity of $\widetilde{O}((\sqrt{\kappa_\ell}/n^{1/4})\log(1/\epsilon))$ for quadratic losses and $\widetilde{O}(\sqrt{\kappa_\ell}(d/n)^{1/4}\log(1/\epsilon))$ for self-concordant losses. Comparing with (5.1.16) and (5.1.17), in both cases they correspond to $O(\sqrt{\kappa_{F/\phi}}\log(1/\epsilon))$ in our notation. Reddi et al. [2016] use the Catalyst framework [Lin et al., 2015a] to accelerate DANE; their method, called AIDE, achieves the same improved complexity for quadratic functions. We obtain similar results for smooth convex functions using direct acceleration.

Yuan and Li [2019] revisited the analysis of DANE and found that the worse complexity of $\widetilde{O}((\kappa_\ell^2/n)\log(1/\epsilon))$ is due to the lost statistical efficiency when averaging m different updates computed by (5.1.11). They propose to use a single local preconditioner at the server and obtain a communication complexity of $\widetilde{O}((1+\kappa_\ell/\sqrt{n})\log(1/\epsilon))$ for quadratic functions. In addition, they propose a variant of DANE with heavy-ball momentum (DANE-HB), and show that it has communication complexity $\widetilde{O}((\sqrt{\kappa_\ell}/n^{1/4})\log(1/\epsilon))$ for quadratic

loss functions, matching that of DiSCO and AIDE. For non-quadratic functions, they show DANE-HB has accelerated local convergence rate near the solution.

Wang et al. [2018b] proposed GIANT, an approximate Newton method that approximates the overall Hessian by the harmonic mean of the local Hessians. It is equivalent to DANE in the quadratic case. They obtain a communication complexity that has logarithmic dependence on the condition number but requires local sample size $n > d$. Mahajan et al. [2018] proposed a distributed algorithm based on local function approximation, which is related to the preconditioning idea of DANE. Wang and Zhang [2019] apply statistical preconditioning to speed up a mini-batch variant of SVRG [Johnson and Zhang, 2013a], but they rely on generic Catalyst acceleration and their convergence results only hold for a very small ball around the optimum.

Distributed optimization methods that use dual variables to coordinate solutions to local subproblems include ADMM [Boyd et al., 2010] and CoCoA [Jaggi et al., 2014, Ma et al., 2015, 2017]. Numerical experiments demonstrate that they benefit from statistical similarities of local functions in the early iterations [Xiao et al., 2019a], but their established communication complexity is no better than $O(\kappa_F \log(1/\epsilon))$.

5.3. The SPAG Algorithm

Although our main motivation in this paper is distributed optimization, the SPAG algorithm works in the general setting of minimizing relatively smooth and strongly convex functions. In this section, we first present SPAG in the more general setting (Algorithm 10), then explain how to run it for distributed empirical risk minimization.

In the general setting, we consider convex optimization problems of the form (5.1.1), where ψ is a closed convex function and F satisfies the following assumption.

ASSUMPTION 9. *F is L_F-smooth and σ_F-strongly convex. In addition, it is $L_{F/\phi}$-smooth and $\sigma_{F/\phi}$-strongly convex with respect to a differentiable convex function ϕ, and ϕ itself is L_ϕ-smooth and σ_ϕ-strongly convex.*

Algorithm 10 requires an initial point $x_0 \in \mathrm{dom}\psi$ and two parameters $L_{F/\phi}$ and $\sigma_{F/\phi}$. During each iteration, Line 6 finds $a_{t+1} > 0$ by solving a quadratic equation, then Line 7 calculates three scalars α_t, β_t and η_t, which are used in the later updates for the three vectors y_t, v_{t+1} and x_{t+1}. The function $V_t(\cdot)$ being minimized in Line 10 is defined as

$$V_t(x) \;=\; \eta_t\Big(\nabla F(y_t)^\top x + \psi(x)\Big) + (1-\beta_t)D_\phi(x, v_t) + \beta_t D_\phi(x, y_t). \tag{5.3.1}$$

The inequality that needs to be satisfied in Line 12 is

$$D_\phi(x_{t+1}, y_t) \le\; \alpha_t^2 G_t\Big((1-\beta_t)D_\phi(v_{t+1}, v_t) + \beta_t D_\phi(v_{t+1}, y_t)\Big), \tag{5.3.2}$$

where G_t is a scaling parameter depending on the properties of D_ϕ. It is a more flexible version of the *triangle scaling gain* introduced by Hanzely et al. [2018].

As we will see in Theorem 26, smaller G_t's correspond to faster convergence rate. Algorithm 10 implements a *gain-search* procedure to automatically find a small G_t. At the beginning of each iteration, the algorithm always trys to set $G_t = G_{t-1}/2$ as long as $G_{t-1} \ge 2$ (G_{t-1} is divided by 4 in Line 3 since it is always multiplied by 2 in Line 5). Whenever (5.3.2) is not satisfied, G_t is multiplied by 2. When the inequality (5.3.2) is satisfied, G_t is within a

Algorithm 10 SPAG$(L_{F/\phi}, \sigma_{\text{rel}}, x_0)$

$v_0 = x_0$, $A_0 = 0$, $B_0 = 1$, $G_{-1} = 1$
for $t = 0, 1, 2, \ldots$ **do**
  $G_t = \max\{1, G_{t-1}/2\}/2$
  repeat
    $G_t \leftarrow 2G_t$
    Find a_{t+1} such that $a_{t+1}^2 L_{\text{rel}} G_t = A_{t+1}B_{t+1}$ where $A_{t+1} = A_t + a_{t+1}$, $B_{t+1} = B_t + a_{t+1}\sigma_{\text{rel}}$
    $\alpha_t = \frac{a_{t+1}}{A_{t+1}}$, $\beta_t = \frac{a_{t+1}}{B_{t+1}}\sigma_{\text{rel}}$, $\eta_t = \frac{a_{t+1}}{B_{t+1}}$
    $y_t = \frac{1}{1-\alpha_t\beta_t}\Big((1-\alpha_t)x_t + \alpha_t(1-\beta_t)v_t\Big)$
    Compute $\nabla F(y_t)$ *(requires communication)*
    $v_{t+1} = \arg\min_x V_t(x)$
    $x_{t+1} = (1-\alpha_t)x_t + \alpha_t v_{t+1}$
  until Inequality (5.3.2) is satisfied

factor of 2 from its smallest possible value. The following lemma guarantees that the gain-search loop always terminates within a small number of steps (see proof in Appendix 5.A).

LEMMA 29. *If Assumption 9 holds, then the inequality* (5.3.2) *holds with* $G_t = \kappa_\phi = L_\phi/\sigma_\phi$.

Therefore, if $\phi = (1/2)\|\cdot\|^2$, then we can set $G_t = 1$ and there is no need to check (5.3.2). In general, Algorithm 10 always produces $G_t < 2\kappa_\phi$ for all $t \geq 0$. Following the argument from Nesterov [2013b, Lemma 4], the total number of gain-searches performed up to iteration t is bounded by

$$2(t+1) + \log_2(G_t),$$

which also bounds the total number of gradient evaluations. Thus the overhead is roughly twice as if there were no gain-search. Next we present a convergence theorem for SPAG.

THEOREM 26. *Suppose Assumption 9 holds. Then the sequences generated by SPAG satisfy for all* $t \geq 0$,

$$\Big(\Phi(x_t) - \Phi(x_*)\Big) + \sigma_{\text{rel}} D_\phi(x_*, v_t) \leq \frac{1}{A_t} D_\phi(x_*, v_0),$$

where $A_t = \frac{1}{4\sigma_{\text{rel}}}\left(\prod_{\tau=0}^{t-1}(1+\gamma_\tau) - \prod_{\tau=0}^{t-1}(1-\gamma_\tau)\right)^2$, *and* $\gamma_t = \frac{1}{2\sqrt{\kappa_{\text{rel}} G_t}}$.

The proof of Theorem 26 relies on the techniques of Nesterov and Stich [2017], and the details are given in Appendix 5.A. We can estimate the convergence rate as follows:

$$\frac{1}{A_t} = O\left(\prod_{\tau=0}^{t}\left(1 - \frac{1}{\sqrt{\kappa_{F/\phi} G_\tau}}\right)\right) = O\left(\left(1 - \frac{1}{\sqrt{\kappa_{F/\phi}\widetilde{G}_t}}\right)^t\right),$$

where $\widetilde{G}_t$ is such that $\widetilde{G}_t^{-1/2} = (1/t)\sum_{\tau=0}^{t} G_t^{-1/2}$, that is, $\widetilde{G}_t^{1/2}$ is the harmonic mean of $G_0^{1/2}, \ldots, G_{t-1}^{1/2}$. In addition, it can be shown that $A_t \geq t^2/(4L_{F/\phi}\widetilde{G}_t)$. Therefore, as $\sigma_{F/\phi} \to 0$,

Theorem 26 gives an accelerated sublinear rate:

$$\Phi(x_t) - \Phi(x_*) \le \frac{4L_{F/\phi}\widetilde{G}_t}{t^2} D_\phi(x_*, x_0).$$

To estimate the worst case when $\sigma_{F/\phi} > 0$, we replace G_t by κ_ϕ to obtain the iteration complexity $O\Big(\sqrt{\kappa_{F/\phi}\kappa_\phi}\log(1/\epsilon)\Big)$. Since $\kappa_{F/\phi}\kappa_\phi \approx \kappa_F$, this is roughly $O\Big(\sqrt{\kappa_F}\log(1/\epsilon)\Big)$, the same as without preconditioning. However, the next lemma shows that under a mild condition, we always have $G_t \to 1$ geometrically.

LEMMA 30. *Suppose Assumption 9 holds and in addition, $\nabla^2\phi$ is M-Lipschitz-continuous, i.e., for all $x, y \in \mathrm{dom}\psi$,*

$$\big\|\nabla^2\phi(x) - \nabla^2\phi(y)\big\| \le M\|x - y\|.$$

Then the inequality (5.3.2) holds with

$$G_t = \min\Big\{\kappa_\phi,\ 1 + (M/\sigma_\phi)d_t\Big\}, \tag{5.3.3}$$

where $d_t = \|v_{t+1} - v_t\| + \|v_{t+1} - y_t\| + \|x_{t+1} - y_t\|$.

In particular, if ϕ is quadratic, then we have $M = 0$ and $G_t = 1$ always satisfies (5.3.2). In this case, the convergence rate in Theorem 26 satisfies $1/A_t = O\Big(\Big(1 - 1/\sqrt{\kappa_{F/\phi}}\Big)^t\Big)$.

In general, $M \neq 0$, but it can be shown that the sequences generated by Algorithm 10, $\{x_t\}$, $\{y_t\}$ and $\{v_t\}$ all converge to x_* at the rate $\Big(1 - 1/\sqrt{\kappa_F}\Big)^t$ [see, e.g., Lin and Xiao, 2015, Theorem 1]. As a result, $d_t \to 0$ and thus $G_t \to 1$ at the same rate. Consequently, the convergence rate established in Theorem 26 quickly approaches $O\Big(\Big(1 - 1/\sqrt{\kappa_{F/\phi}}\Big)^t\Big)$.

5.3.1. Implementation for Distributed Optimization. In distributed optimization, Algorithm 10 is implemented at the server. During each iteration, communication between the server and the workers only happens when computing $\nabla F(y_t)$. Checking if the inequality (5.3.2) holds locally requires that the server has access to the preconditioner ϕ.

If the datasets on different workers are i.i.d. samples from the same source distribution, then we can use any f_j in the definition of ϕ in (5.1.10) and assign worker j as the server. However, this is often not the case in practice and obtaining i.i.d. datasets on different workers may involve expensive shuffling and exchanging large amount of data among the workers. In this case, a better alternative is to randomly sample small portions of the data on each worker and send them to a dedicated server. We call this sub-sampled dataset $\mathcal{D}_0$ and the local loss at the server f_0, which is defined the same way as in (5.1.3). Then the server implements Algorithm 10 with $\phi(x) = f_0(x) + (\mu/2)\|x\|^2$. Here we only need $\mathcal{D}_0$ be a uniform sub-sample of $\cup_{j=1}^m \mathcal{D}_j$, which is critical for effective preconditioning. On the other hand, it is not a problem at all if the datasets at the workers, $\mathcal{D}_1, \ldots, \mathcal{D}_m$, are not shuffled to to be i.i.d., because it does not change the average gradients $\nabla F(y_t)$. In the rest of the paper, we omit the subscript to simply use f to represent the local empirical loss function. As discussed in Section 5.1.1, if

$$\big\|\nabla^2 f(x) - \nabla^2 F(x)\big\| \le \mu, \quad \forall\, x \in \mathrm{dom}\psi \tag{5.3.4}$$

with high probability, then according to (5.1.14), we can choose

$$L_{F/\phi} = 1, \qquad \sigma_{F/\phi} = \frac{\sigma_F}{\sigma_F + 2\mu}$$

as the input to Algorithm 10. In the next section, we leverage matrix concentration bounds to estimate how μ varies with the number of subsamples n. With sufficiently large n, we can make μ small so that the relative condition number $\kappa_{F/\phi} = 1 + 2\mu/\sigma_F$ is much smaller than κ_F.

5.4. Bounding the Relative Condition Number

In this section, we derive refined matrix concentration bounds for linear prediction models. Suppose the overall dataset consists of N samples $\{z_1, \ldots, z_N\}$, where each $z_i = (a_i, b_i)$ with $a_i \in \mathbb{R}^d$ being a feature vector and b_i the corresponding label or regression target. Linear models (including logistic and ridge regression) have the form $\ell(x, z_i) = \ell_i(a_i^\top x) + \frac{\lambda}{2}\|x\|^2$, where ℓ_i is twice differentiable and may depend on b_i, and $\lambda > 0$. We further assume that $\ell_i'' = \ell_j''$ for all i and j, which is valid for logistic and ridge regression as well. Since $f(x) = (1/n)\sum_{i=1}^n \ell(x, z_i)$, we have

$$\nabla^2 f(x) = \frac{1}{n}\sum_{i=1}^n \ell_i''(a_i^\top x) a_i a_i^\top + \lambda I_d. \tag{5.4.1}$$

Here we omit the subscript j in f_j since we only need one subsampled dataset at the server, as explained in Section 5.3.1. For the overall loss function defined in (5.1.2), the Hessian $\nabla^2 F(x)$ is defined similarly by replacing n with N.

We assume for simplicity that the strong convexity of F mainly comes from regularization, that is, $\sigma_F = \sigma_\ell = \lambda$, but the results can be easily extended to account for the strong convexity from data. We start by showing tight results for quadratics, and then provide uniform concentration bounds of Hessians for more general loss functions. Finally, we give a refined bound when the a_i's are sub-Gaussian.

5.4.1. Quadratic Case. We assume in this section that $\ell_i(a_i^\top x) = (a_i^\top x - b_i)^2/2$, and that there exists a constant R such that $\|a_i\| \le R$ for all $i = 1, \ldots, N$. In this case we have $L_\ell = R^2$ and $\kappa_\ell = R^2/\lambda$. Since the Hessians do not depend on x, we use the notation

$$H_F = \nabla^2 F(x), \quad H_f = \nabla^2 f(x).$$

Previous works [Shamir et al., 2014, Reddi et al., 2016, Yuan and Li, 2019] use the Hoeffding bound (5.1.15) to obtain

$$\left(1 + \frac{2\mu}{\lambda}\right)^{-1} (H_f + \mu I_d) \preceq H_F \preceq H_f + \mu I_d, \tag{5.4.2}$$

$$\text{with} \qquad \mu = \frac{R^2}{\sqrt{n}}\sqrt{32\log(d/\delta)}. \tag{5.4.3}$$

Our result is given in the following theorem.

THEOREM 27. *Suppose ℓ_i is quadratic and $\|a_i\| \le R$ for all i. For a fixed $\delta > 0$, if $n > \frac{28}{3}\log\left(\frac{2d}{\delta}\right)$, then the following inequality holds with probability at least $1 - \delta$:*

$$\left(\frac{3}{2} + \frac{2\mu}{\lambda}\right)^{-1} (H_f + \mu I_d) \preceq H_F \preceq 2\,(H_f + \mu I_d)\,, \tag{5.4.4}$$

$$\textit{with} \qquad \mu = \frac{1}{2}\left(\frac{28R^2}{3n}\log\left(\frac{2d}{\delta}\right) - \lambda\right)^+. \tag{5.4.5}$$

Thus, for this choice of μ, $\sigma_{\text{rel}} = \left(\frac{3}{2} + \frac{2\mu}{\lambda}\right)^{-1}$, $L_{\text{rel}} = 2$ *and so* $\kappa_{\text{rel}} = O\left(1 + \frac{\kappa_\ell}{n}\log\left(\frac{d}{\delta}\right)\right)$ *with probability* $1-\delta$.

Theorem 27 improves on the result in (5.4.3) by a factor of $\sqrt{n}$. The reason is that matrix inequality (5.4.2) is derived from the additive bound $\|H_f - H_F\| \leq \mu$ [e.g., Shamir et al., 2014, Yuan and Li, 2019]. We derive the matrix inequality (5.4.4) directly from a multiplicative bound using the matrix Bernstein inequality (see proof in Appendix 5.B.1). Note that by using matrix Bernstein instead of matrix Hoeffding inequality [Tropp, 2015], one can refine the bound for μ in (5.4.2) from $L_\ell/\sqrt{n}$ to $\sqrt{L_\ell L_F/n}$, which can be as small as L_ℓ/n in the extreme case when all the a_i's are orthogonal. Our bound in (5.4.5) states that $\mu = \widetilde{O}(L_\ell/n)$ in general for quadratic problems, leading to $\kappa_{F/\phi} = \widetilde{O}(1+\kappa_\ell/n)$.

Remark 1. Theorem 27 is proved by assuming random sampling with replacement. In practice, we mostly use random sampling without replacement, which usually concentrates even more than with replacement [Hoeffding, 1963].

Remark 2. In terms of reducing $\kappa_{F/\phi}$, there is not much benefit to having $\mu < \lambda$. Indeed, higher values of μ regularize the inner problem of minimizing $V_t(x)$ in (5.3.1), because the condition number of $D_\phi(x,y) = D_f(x,y) + (\mu/2)\|x-y\|^2$ is $(L_f+\mu)/(\lambda+\mu)$. Increasing μ can thus lead to substantially easier subproblems when $\mu > \lambda$, which reduces the computation cost at the server, although this may sometimes affect the rate of convergence.

5.4.2. Non-quadratic Case. For non-quadratic loss functions, we need $\nabla^2 f(x)$ to be a good approximation of $\nabla^2 F(x)$ for all iterations of the SPAG algorithm. It is tempting to argue that concentration only needs to hold for the iterates of SPAG, and a union bound would then give an extra $\log T$ factors for T iterations. Yet this only works for one step since x_t depends on the points chosen to build f for $t > 0$, so the $\ell''(a_i^\top x_t)a_i a_i^\top$ are not independent for different i (because of x_t). Therefore, the concentration bounds need to be written at points that do not depend on f. In order to achieve this, we restrict the optimization variable within a bounded convex set and prove uniform concentration of Hessians over the set. Without loss of generality, we consider optimization problems constrained in $\mathcal{B}(0,D)$, the ball of radius D centered at 0. Correspondingly, we set the nonsmooth regularization function as $\psi(x) = 0$ if $x \in \mathcal{B}(0,D)$ and infinity otherwise.

If the radius D is small, it is then possible to leverage the quadratic bound by using the inequality

$$\begin{aligned}\|H_f(x) - H_F(x)\| \;\leq\; &\|H_f(x) - H_f(y)\| \\ &+ \|H_f(y) - H_F(y)\| + \|H_F(x) - H_F(y)\|.\end{aligned}$$

Thus, under a Lipschitz-continuous Hessian assumption (which we have), only concentration at point y matters. Yet, such bounding is only meaningful when x is close to y, thus leading to the very small convergence radius of Wang and Zhang [2019, Theorem 13], in which they use concentration at the optimal point x_*. Using this argument for several y's that pave $\mathcal{B}(0,D)$ leads to an extra $\sqrt{d}$ multiplicative factor since concentration needs to hold at exponentially (in d) many points, as discussed in Section 5.1.1. We take a different approach

in this work, and proceed by directly bounding the supremum for all $x \in \mathcal{B}(0, D)$, thus looking for the smallest μ that satisfies:

$$\sup_{x \in \mathcal{B}(0,D)} \|H_f(x) - H_F(x)\|_{\text{op}} \leq \mu. \tag{5.4.6}$$

Equation (5.4.2) can then be used with this specific μ. We now introduce Assumption 10, which is for example verified for logistic regression with $B_\ell = 1/4$ and $M_\ell = 1$.

ASSUMPTION 10. *There exist B_ℓ and M_ℓ such that ℓ_i'' is M_ℓ-Lipschitz continuous and $0 \leq \ell_i''(a^\top x) \leq B_\ell$ almost surely for all $x \in \mathcal{B}(0, D)$.*

THEOREM 28. *If ℓ_i satisfies Assumption 10, then Equation (5.4.6) is satisfied with probability at least $1 - \delta$ for*

$$\mu = \sqrt{4\pi} \frac{R^2}{\sqrt{n}} \left(B_\ell \left[2 + \sqrt{\frac{1}{2\pi} \log(\delta^{-1})} \right] + R M_\ell D \right).$$

SKETCH OF PROOF. The high probability bound on the supremum is obtained using Mc Diarmid inequality [Boucheron et al., 2013]. This requires a bound on its expectation, which is obtained using symmetrization and the Sudakov-Fernique Lemma [Boucheron et al., 2013]. The complete proof can be found in Appendix 5.B.2. □

The bound of Theorem 28 is relatively tight as long as $RM_\ell D < B_\ell \sqrt{\log(\delta^{-1})}$. Indeed, using the matrix Bernstein inequality for a fixed $x \in \mathcal{B}(0, D)$ would yield $\mu = O\left(R\sqrt{L_F} B_\ell \log(d/\delta)/\sqrt{n}\right)$. Therefore, Theorem 28 is tight up to a factor $R/\sqrt{L_F}$ in this case.

5.4.3. Sub-Gaussian Bound. We show in this section that the bound of Theorem 28 can be improved under a stronger sub-Gaussian assumption on a.

DEFINITION 6. *The random variable $a \in \mathbb{R}^d$ is sub-Gaussian with parameter $\rho > 0$ if one has for all $\epsilon > 0$, $x \in \mathcal{B}(0, D)$:*

$$\mathbb{P}(|a_i^\top x)| \geq \rho\epsilon) \leq 2e^{-\frac{\epsilon^2}{2\|x\|^2}}. \tag{5.4.7}$$

THEOREM 29. *If ℓ_i satisfies Assumption 10 and the a_i are sub-Gaussian with constant ρ, then denoting $\tilde{B} = B_\ell/(M_\ell D)$, there exists $C > 0$ such that Equation (5.4.6) is satisfied with probability $1 - \delta$ for*

$$\mu = C \frac{\rho^2 M_\ell D}{\sqrt{n}} (d + \log(\delta^{-1})) \left[\frac{\rho + \tilde{B}}{\sqrt{d}} + \frac{\rho + (R^2 \tilde{B})^{\frac{1}{3}}}{\sqrt{n}} \right].$$

SKETCH OF PROOF. This bound is a specific instantiation of a more general result based on chaining, which is a standard argument for proving results on suprema of empirical processes [Boucheron et al., 2013]. The complete proof can be found in Appendix 5.B.3. □

The sub-Gaussian assumption (5.4.7) always holds with $\rho = R$, the almost sure bound on $\|a_i\|$. However Theorem 29 improves over Theorem 28 only with a stronger sub-Gaussian assumption, i.e., when $\rho < R$. In particular for a_i uniform over $\mathcal{B}(0, R)$, one has $\rho = R/\sqrt{d}$. Assuming further that the $(R^2 B)^{1/3}/\sqrt{n}$ term dominates yields $\mu = O(R^2 (R^2 B)^{1/3}/n)$, a

$\sqrt{n}$ improvement over Theorem 28. We expect tighter versions of Theorem 29, involving the effective dimension d_{eff} of vectors a_i instead of the full dimension d, to hold.

5.5. Experiments

We have seen in the previous section that preconditioned gradient methods can outperform gradient descent by a large margin in terms of communication rounds, which was already observed empirically [Shamir et al., 2014, Reddi et al., 2016, Yuan and Li, 2019]. We compare in this section the performances of SPAG with those of DANE and its heavy-ball acceleration, HB-DANE [Yuan and Li, 2019], as well as accelerated gradient descent (AGD). For this, we use two datasets from LibSVM[1], RCV1 Lewis et al. [2004] and the preprocessed version of KDD2010 (algebra) Yu et al. [2010]. Due to its better convergence guarantees [Shamir et al., 2014, Yuan and Li, 2019], DANE refers in this section to the proximal gradient method with the Bregman divergence associated to $\phi = f_1 + (\mu/2)\|\cdot\|^2$ (without averaging over m workers). We use SPAG with $\sigma_{\text{rel}} = 1/(1+2\mu/\lambda)$ and HB-DANE with $\beta = (1-(1+2\mu/\lambda)^{-1/2})^2$. Fine tuning these parameters only leads to comparable small improvements for both algorithms. We tune both the learning rate and the momentum of AGD.

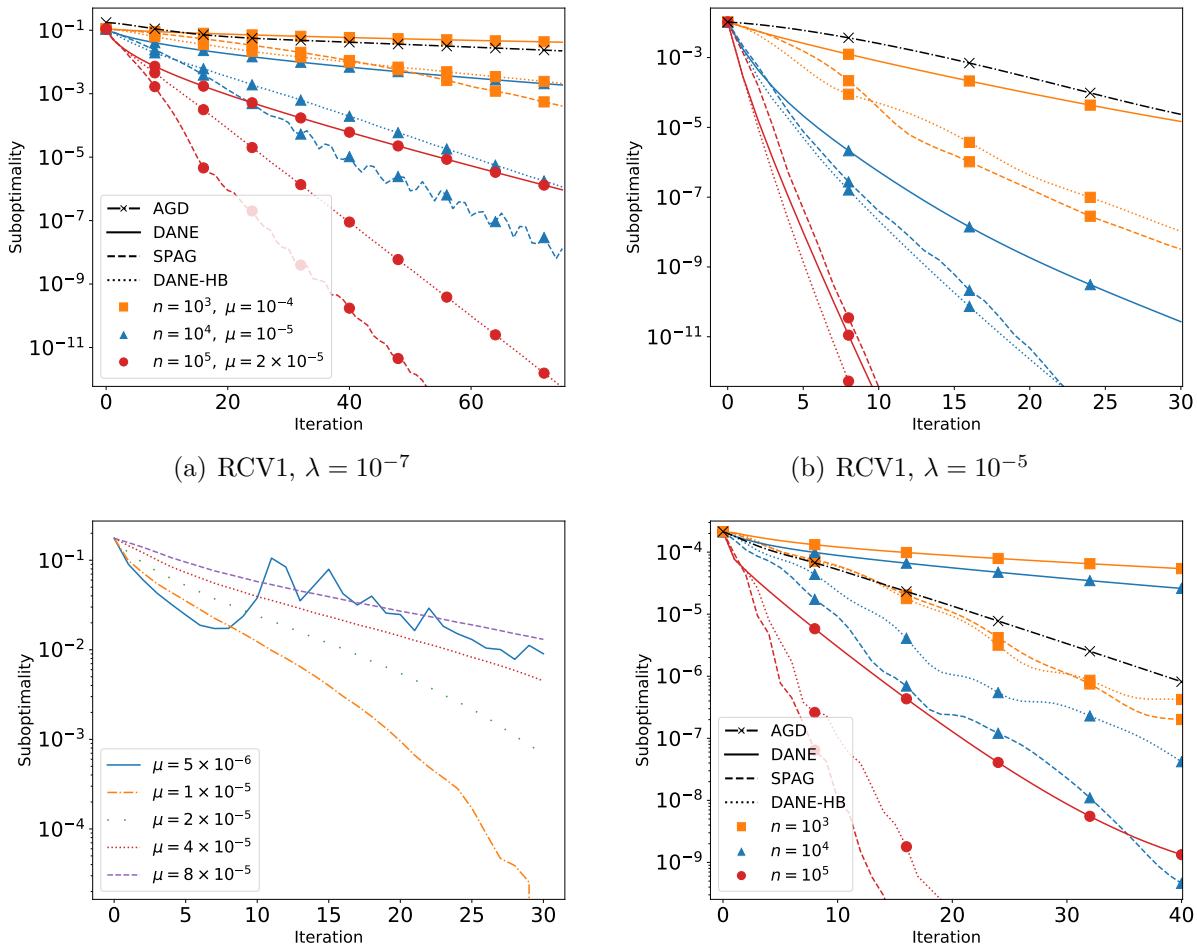
Note that, as mentioned in Section 5.3.1, the number of nodes used by SPAG does not affect its iteration complexity (but change the parallelism of computing $\nabla F(x_t)$). Only the size n of the dataset used for preconditioning matters. We initialize all algorithms at the same point, which is the minimizer of the server's entire local loss (regardless of how many samples are used for preconditioning).

Tuning μ. Although μ can be estimated using concentrations results, as done in Section 5.4, these bounds are too loose to be used in practice. Yet, they show that μ depends very weakly on λ. This is verified experimentally, and we therefore use the same value for μ regardless of λ. To test the impact of μ on the iteration complexity, we fix a step-size of 1 and plot the convergence speed of SPAG for several values of μ. We see on Figure 1(c) that the value of μ drastically affects convergence, actually playing a role similar to the inverse of a step-size. Indeed, the smaller the μ the faster the convergence, up to a point at which the algorithm is not stable anymore. Convergence could be obtained for smaller values of μ by taking a smaller step-size. Yet, the step-size needs to be tuned for each value of μ, and we observed that this does not lead to significant improvements in practice. Thus, we stick to the guidelines for DANE by Shamir et al. [2014], i.e., we choose $L_{\text{rel}} = 1$ and tune μ.

Line search for G_t. As explained in Section 5.3, the optimal G_t is obtained through a line search. Yet, we observed in all our experiments that $G_t = 1$ most of the time. This is due to the fact that we start at the minimizer of the local cost function, which can be close to the global solution. In addition, Equation (5.3.3) can actually be verified for $G_t < 1$, even in the quadratic. Therefore, the line search generally has no added cost (apart from checking that $G_t = 1$ works) and the effective rate in our experiments is $\kappa_{\text{rel}}^{-1/2}$. Experiments displayed in Figure 1 use $G_t = 1$ for simplicity.

RCV1. Figures 1(b) and 1(a) present results for the RCV1 dataset with different regularizations. All algorithms are run with $N = 677399$ (split over 4 nodes) and $d = 47236$. We see that in Figure 1(b), the curves can be clustered by values of n, meaning that when

[1]Accessible at https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/binary.html



(a) RCV1, $\lambda = 10^{-7}$

(b) RCV1, $\lambda = 10^{-5}$

(c) Effect of μ on the convergence speed of SPAG on RCV1 with $\lambda = 10^{-7}$ and $n = 10^4$.

(d) KDD2010, $\lambda = 10^{-7}$. We use $\mu = 0.1/(2n)$, except for $n = 10^3$, where $\mu = 10^{-5}$, and $L_{\text{rel}} = 2$ for SPAG.

FIGURE 1. Experimental results

regularization is relatively high ($\lambda = 10^{-5}$), increasing the preconditioning sample size has a greater effect than acceleration since the problem is already well-conditioned. In particular, acceleration does not improve the convergence rate when $n = 10^5$ and $\lambda = 10^{-5}$. When regularization is smaller ($\lambda = 10^{-7}$), SPAG and HB-DANE outperform DANE even when ten times less samples are used for preconditioning, as shown in Figure 1(a). Finer tuning (without using the theoretical parameters) of the momentum marginally improves the performances of SPAG and HB-DANE, at the cost of a grid search. SPAG generally outperforms HB-DANE in our experiments, but both methods have comparable asymptotic rates.

KDD2010. Figure 1(d) presents the results of larger scale experiments on a random subset of the KDD2010 dataset with $N = 7557074$ (split over 80 nodes), $d = 20216830$ and $\lambda = 10^{-7}$. The conclusions are similar to the experiments on RCV1, i.e., acceleration allows to use significantly less samples at the server for a given convergence speed. AGD competes with DANE when λ and n are small, but it is outperformed by SPAG in all our

experiments. More experiments investigating the impact of line search, tuning and inaccurate local solutions are presented in Appendix 5.C.

5.6. Conclusion

We have introduced SPAG, an accelerated algorithm that performs statistical preconditioning for large-scale optimization. Although our motivation in this paper is for distributed empirical risk minimization, SPAG applies to much more general settings. We have given tight bounds on the relative condition number, a crucial quantity to understand the convergence rate of preconditioned algorithms. We have also shown, both in theory and in experiments, that acceleration allows SPAG to efficiently leverage rough preconditioning with limited number of local samples. Preliminary experiments suggest that SPAG is more robust to inaccurate solution of the inner problems than HB-DANE. Characterizing the effects of inaccurate inner solutions in the preconditioning setting would be an interesting extension of this work.

5.7. Statistical Preconditioning with other Bregman algorithms

In this Section, we study statistical preconditioning using an accelerated Bregman method to improve the dependence on the relative condition number. Yet, other Bregman methods can be considered, such as Bregman SGD. This section presents results from Dragomir, Even, and Hendrikx [2021a], that can be applied in the context of distributed optimization in the same way as it is done in Chapter 5. More specifically, we consider the same setting as Theorem 5, but this time the iterations are the following:

$$\nabla h(x_{t+1}) = \nabla h(x_t) - \eta g_t, \tag{5.7.1}$$

Section 4.A introduces a Bregman coordinate gradient algorithm, in which $g_t = e_i e_i^\top \nabla f(x_t)$, that has a distributed interpretation. In this section, we consider a general Bregman SGD algotithm, and we make the following assumptions on the stochastic gradients g_t.

Assumption 11. *The stochastic gradients g_t are such that $g_t = \nabla f_{\xi_t}(x_t)$, with $\mathbb{E}_{\xi_t}[f_{\xi_t}] = f$ and f_{ξ_t} is $L_{f/h}$-relatively smooth with respect to h for all ξ_t. Besides, there exists a constant $\chi^2 \geq 0$ such that:*

$$\begin{aligned}\chi^2 &\geq \frac{1}{2\eta^2}\mathbb{E}_{\xi_t}\left[D_{h^*}(\nabla h(x_t) - 2\eta\nabla f_{\xi_t}(x^\star), \nabla h(x_t))\right] \\ &= \mathbb{E}_{\xi_t}\left[\|\nabla f_{\xi_t}(x^\star)\|^2_{\nabla^2 h^*(z_t)}\right],\end{aligned}$$

for some $z_t \in [\nabla h(x_t) - 2\eta\nabla f_{\xi_t}(x^\star), \nabla h(x_t)]$.

The definition of the variance χ is a Bregman adaptation of the usual variance at the optimum definition used for instance in Bach and Moulines [2011b]. Note that since h^* is $1/\sigma_h$-smooth, then the assumption is verified for instance when the variance is bounded in Euclidean norm, as $\|\nabla f_{\xi_t}(x^*)\|^2_{\nabla^2 h^*(z_t)} \leq \sigma_h^{-1}\|\nabla f_{\xi_t}(x^*)\|^2$. We can now prove the following Theorem:

Theorem 30. *If f is L-smooth and σ-strongly convex relative to h with $\sigma > 0$, and Assumption 11 hold, then for $\eta \leq 1/(2L_{\text{rel}})$, the iterates produced by Bregman stochastic*

gradient (5.7.1) *satisfy*

$$\mathbb{E}[D_h(x^\star, x_t)] \leq (1-\eta\mu)^t D_h(x^\star, x_0) + \eta\frac{\sigma^2}{\mu}. \tag{5.7.2}$$

The proof is based on the two following lemmas. We omit the proofs in this thesis, and refer the interested reader to Dragomir et al. [2021a]. Lemma 31 is a Bregman equivalent of the celebrated inequality $2ab \leq a^2 + b^2$, and Lemma 32 is a generalization of the cocoercivity of the gradients [Nesterov, 2013c, Eq. 2.1.7] to the relatively smooth case.

LEMMA 31. *Let x^+ be such that $\nabla h(x^+) = \nabla h(x) - g$, and similarly define x_1^+ and x_2^+ from g_1 and g_2. Then, if $g = \frac{g_1+g_2}{2}$, we obtain:*

$$D_h(x, x^+) \leq \frac{1}{2}\left[D_h(x, x_1^+) + D_h(x, x_2^+)\right].$$

LEMMA 32 (Bregman Cocoercivity). *If a convex function f is relatively L-smooth w.r.t to h, then for any $\eta \leq \frac{1}{L}$,*

$$D_f(x, y) \geq \frac{1}{\eta} D_{h^*}\left(\nabla h(x) - \eta\left(\nabla f(x) - \nabla f(y)\right), \nabla h(x)\right) \tag{5.7.3}$$

PROOF OF THEOREM 30. Since the gradient is unbiased ($\mathbb{E}[g_t] = \nabla f(x_t)$), we obtain similarly to Equation (1.1.44) that:

$$\mathbb{E}[D_h(x^\star, x_{t+1})] = D_h(x^\star, x_t) - \eta D_f(x^\star, x_t) - \eta D_f(x_t, x^\star) + \mathbb{E}[D_h(x_t, x_{t+1})]. \tag{5.7.4}$$

Using Lemma 31, the last term can be bounded as $D_h(x_t, x_{t+1}) \leq \frac{1}{2}\left[D_1 + D_2\right]$. We use Lemma 32 (Bregman co-coercivity) to write:

$$D_1 = D_{h^*}(\nabla h(x_t) - 2\eta\left[\nabla f_{\xi_t}(x_t) - \nabla f_{\xi_t}(x^\star)\right], \nabla h(x_t)) \leq 2\eta D_{f_{\xi_t}}(x_t, x^\star),$$

so that $\mathbb{E}[D_1/2] \leq \eta D_f(x_t, x^\star)$. Similarly,

$$D_2 = D_{h^*}(\nabla h(x_t) - 2\eta\nabla f_\xi(x^\star), \nabla h(x_t)), \tag{5.7.5}$$

so that $\mathbb{E}[D_2/2] \leq \eta^2\chi^2$. Thus, using the relative strong convexity of f to bound the $D_f(x^\star, x_t)$ term, we obtain:

$$\mathbb{E}[D_h(x^\star, x_{t+1})] \leq (1-\eta\sigma) D_h(x^\star, x_t) + \eta^2\chi^2, \tag{5.7.6}$$

which yields the desired result. □

Theorem 30 in particular states that in the interpolation setting (the variance at the optimum $\chi = 0$) then Bregman SGD is as fast as Bregman GD, just like in the Euclidean case. Besides, a convex formulation of Theorem 30 can be stated (when $\sigma_{\text{rel}} = 0$), but we refer the interested reader to Dragomir et al. [2021a] for more details.

In Dragomir et al. [2021a], we then present a variance-reduced version of the iterations (5.7.1), with an asymptotic convergence rate that matches the stochastic speedup discussed in Section 1.2, though at the cost of additional regularity assumptions. Since we do not present distributed interpretation of this method in this thesis (although they can be used in the context of distributed optimization in the same way as in Chapter 5), we refer the reader to the corresponding paper for the specifics of variance-reduced Bregman stochastic methods.

Appendix

5.A. Convergence Analysis of SPAG

This section provides proofs for Lemma 29, Theorem 26 and Lemma 30 presented in Section 5.3. Before getting to the proofs, we first comment on the nature of the accelerated convergence rate obtained in Theorem 26.

Note that SPAG (Algorithm 10) can be considered as an accelerated variant of the general mirror descent method considered by Bauschke et al. [2017] and Lu et al. [2018]. Specifically, we can replace D_ϕ by the Bregman divergence of any convex function of Legendre type [Rockafellar, 1970, Section 26]. Recently, Dragomir et al. [2019] show that fully accelerated convergence rates, as those for Euclidean mirror-maps achieved by Nesterov [2004], may not be attainable in the general setting. However, this negative result does not prevent us from obtaining better accelerated rates in the preconditioned setting. Indeed, we choose a smooth and strongly convex mirror map and further assume Lipschitz continuity of its Hessian. For smooth and strongly convex cost functions, the convergence rates of SPAG are almost always better than those obtained by standard accelerated algorithms (without preconditioning) as long as n is not too small, and can be much better with a good preconditioner.

5.A.1. Proof of Lemma 29. Using the second-order Taylor expansion (mean-value theorem), we have

$$D_\phi(x,y) = \phi(x) - \phi(y) - \langle \nabla\phi(y), x-y\rangle = \frac{1}{2}(x-y)^\top \nabla^2\phi\Big(y + t(x-y)\Big)(x-y),$$

for some scalar $t \in [0,1]$. We define

$$H(x,y) = \nabla^2\phi\Big(y + t(x-y)\Big),$$

where the dependence on $t \in [0,1]$ is made implicit with the ordered pair (x,y). Then we can write

$$D_\phi(x,y) = \frac{1}{2}\|x-y\|^2_{H(x,y)}.$$

By Assumption 9, ϕ is L_ϕ-smooth and σ_ϕ-strongly convex, which implies that for all $x, y \in \mathbb{R}^d$,

$$\sigma_\phi\|x-y\|^2 \le \|x-y\|^2_{H(x,y)} \le L_\phi\|x-y\|^2.$$

Let $w_t = (1-\beta_t)v_t + \beta_t y_t$. Then we have $x_{t+1} - y_t = \alpha_t\Big(v_{t+1} - w_t\Big)$ and

$$\begin{aligned} D_\phi(x_{t+1}, y_t) &= \frac{1}{2}\|x_{t+1} - y_t\|^2_{H(x_{t+1},y_t)} \\ &\le \frac{L_\phi}{2}\|x_{t+1} - y_t\|^2 = \alpha_t^2\frac{L_\phi}{2}\|v_{t+1} - w_t\|^2. \end{aligned}$$

Next we use $v_{t+1} - w_t = (1-\beta_t)(v_{t+1} - v_t) + \beta_t(v_{t+1} - y_t)$ and convexity of $\|\cdot\|^2$ to obtain

$$\begin{aligned} D_\phi(x_{t+1}, y_t) &\le \alpha_t^2\frac{L_\phi}{2}\Big((1-\beta_t)\|v_{t+1} - v_t\|^2 + \beta_t\|v_{t+1} - y_t\|^2\Big) \\ &\le \alpha_t^2\frac{L_\phi}{2\sigma_\phi}\Big((1-\beta_t)\|v_{t+1} - v_t\|^2_{H(v_{t+1},v_t)} + \beta_t\|v_{t+1} - y_t\|^2_{H(v_{t+1},y_t)}\Big) \end{aligned}$$

$$= \alpha_t^2 \kappa_\phi \Big((1-\beta_t) D_\phi(v_{t+1}, v_t) + \beta_t D_\phi(v_{t+1}, y_t) \Big).$$

This finishes the proof of Lemma 29.

5.A.2. Proof of Theorem 26. Theorem 2 is a direct consequence of the following result, which is adapted from Nesterov and Stich [2017].

THEOREM 31 (Smooth and strongly convex mirror map ϕ). *Suppose Assumption 9 holds. Then the sequences generated by Algorithm 10 satisfy for all $t \geq 0$,*

$$A_t \Big(\Phi(x_t) - \Phi(x_*) \Big) + B_t D(x_*, v_t) \leq A_0 \Big(F(x_0) - F(x_*) \Big) + B_0 D(x_*, v_0).$$

Moreover, if we set $A_0 = 0$ and $B_0 = 1$ then for $t \geq 0$,

$$A_t \geq \frac{1}{4\sigma_{\text{rel}}} \left[\pi_t^+ - \pi_t^- \right]^2, \qquad B_t = 1 + \sigma_{\text{rel}} A_t \geq \frac{1}{4} \left[\pi_t^+ + \pi_t^- \right]^2,$$

where

$$\pi_t^+ = \prod_{i=0}^{t-1} \left(1 + \sqrt{\frac{\sigma_{\text{rel}}}{L_{\text{rel}} G_t}} \right), \qquad \pi_t^- = \prod_{i=0}^{t-1} \left(1 - \sqrt{\frac{\sigma_{\text{rel}}}{L_{\text{rel}} G_t}} \right).$$

We first state an equivalent definition of relative smoothness and relative strong convexity [Lu et al., 2018]. The function F is said to be $L_{F/\phi}$-smooth and $\sigma_{F/\phi}$-strongly convex with respect to ϕ if for all $x, y \in \mathbb{R}^d$,

$$F(y) + \nabla F(y)^\top (x - y) + \sigma_{L/\phi} D_\phi(x, y) \;\leq\; F(x) \;\leq\; F(y) + \nabla F(y)^\top (x - y) + L_{L/\phi} D_\phi(x, y). \tag{5.A.1}$$

Obviously this is the same as (5.1.9). We also need the following lemma, which is an extension of a result from Chen and Teboulle [1993, Lemma 3.2], whose proof we omit.

LEMMA 33 (Descent property of Bregman proximal point). *Suppose g is a convex function defined over* domϕ *and*

$$v_{t+1} = \operatorname*{argmin}_x \Big\{ g(x) + (1-\beta_t) D_\phi(x, v_t) + \beta_t D_\phi(x, y_t) \Big\},$$

then for any $x \in$ dom$h$,

$$g(v_{t+1}) + (1-\beta_t) D_\phi(v_{t+1}, v_t) + \beta_t D_\phi(v_{t+1}, y_t) \leq g(x) + (1-\beta_t) D_\phi(x, v_t) + \beta_t D_\phi(x, y_t) - D_\phi(x, v_{t+1}).$$

PROOF OF THEOREM 31. The proof follows the same lines as Nesterov and Stich [2017], with adaptations to use general Bregman divergences. Applying Lemma 33 with $g(x) = \eta_t \Big(\nabla f(y_t)^\top x + \psi(x) \Big)$, we have for any $x \in$ domϕ,

$$\begin{aligned} &D(x, v_{t+1}) + (1-\beta_t) D(v_{t+1}, v_t) + \beta_t D(v_{t+1}, y_t) - (1-\beta_t) D(x, v_t) - \beta_t D(x, y_t) \\ &\leq \eta_t \nabla f(y_t)^\top (x - v_{t+1}) + \eta_t \Big(\psi(x) - \psi(v_{t+1}) \Big). \end{aligned}$$

Since by definition $\eta_t = \frac{a_{t+1}}{B_{t+1}}$, multiplying both sides of the above inequality by B_{t+1} yields

$$\begin{aligned} &B_{t+1} D(x, v_{t+1}) + B_{t+1} \Big((1-\beta_t) D(v_{t+1}, v_t) + \beta_t D(v_{t+1}, y_t) \Big) - B_{t+1} (1-\beta_t) D(x, v_t) \\ &- B_{t+1} \beta_t D(x, y_t) \leq a_{t+1} \nabla f(y_t)^\top (x - v_{t+1}) + a_{t+1} \Big(\psi(x) - \psi(v_{t+1}) \Big). \end{aligned}$$

Using the scaling property (5.3.2) and the relationships $\alpha_t = \frac{a_{t+1}}{A_{t+1}}$ and $a_{t+1}^2 L_{f/\phi} G_t = A_{t+1} B_{t+1}$, we obtain

$$B_{t+1}\Big((1-\beta_t)D(v_{t+1},v_t)+\beta_t D(v_{t+1},y_t)\Big) \geq \frac{B_{t+1}}{\alpha_t^2 G_t} D(x_{t+1},y_t) = \frac{A_{t+1}^2 B_{t+1}}{a_{t+1}^2 G_t} D(x_{t+1},y_t)$$
$$= A_{t+1} L_{f/\phi} D(x_{t+1},y_t).$$

Combining the last two inequalities and using the facts $B_{t+1}(1-\beta_t) = B_t$ and $B_{t+1}\beta_t = a_{t+1}\sigma_{f/\phi}$, we arrive at

$$\begin{aligned} &B_{t+1}D(x,v_{t+1}) + A_{t+1}L_{f/\phi}D(x_{t+1},y_t) - B_t D(x,v_t) - a_{t+1}\sigma_{f/\phi}D(x,y_t) \\ &\leq a_{t+1}\nabla f(y_t)^\top (x - v_{t+1}) + a_{t+1}\Big(\psi(x) - \psi(v_{t+1})\Big). \end{aligned} \tag{5.A.2}$$

We then expand the gradient term on the right-hand side of (5.A.2) into two parts:

$$a_{t+1}\nabla f(y_t)^\top (x - v_{t+1}) = a_{t+1}\nabla f(y_t)^\top (x - w_t) + a_{t+1}\nabla f(y_t)^\top (w_t - v_{t+1}), \tag{5.A.3}$$

where $w_t = (1-\beta_t)v_t + \beta_t y_t$. For the first part,

$$\begin{aligned} a_{t+1}\nabla f(y_t)^\top (x - w_t) &= a_{t+1}\nabla f(y_t)^\top (x - y_t) + \frac{a_{t+1}(1-\alpha_t)}{\alpha_t}\nabla f(y_t)^\top (x_t - y_t) \\ &\leq a_{t+1}\left(f(x) - f(y_t) - \sigma_{f/\phi}D(x,y_t)\right) + \frac{a_{t+1}(1-\alpha_t)}{\alpha_t}\left(f(x_t) - f(y_t)\right). \end{aligned} \tag{5.A.4}$$

Notice that

$$a_{t+1}\frac{1-\alpha_t}{\alpha_t} = a_{t+1}\left(\frac{1}{\alpha_t} - 1\right) = a_{t+1}\left(\frac{A_{t+1}}{a_{t+1}} - 1\right) = A_{t+1} - a_{t+1} = A_t.$$

Therefore, Equation (5.A.4) becomes

$$a_{t+1}\nabla f(y_t)^\top (x - w_t) \leq a_{t+1}f(x) - A_{t+1}f(y_t) + A_t f(x_t) - a_{t+1}\sigma_{f/\phi}D(x,y_t). \tag{5.A.5}$$

For the second part on the right-hand side of (5.A.3),

$$\begin{aligned} &a_{t+1}\nabla f(y_t)^\top (w_t - v_{t+1}) = -\frac{a_{t+1}}{\alpha_t}\nabla f(y_t)^\top (x_{t+1} - y_t) = -A_{t+1}\nabla f(y_t)^\top (x_{t+1} - y_t) \\ &\leq -A_{t+1}\left(f(x_{t+1}) - f(y_t) - L_{f/\phi}D(x_{t+1},y_t)\right), \end{aligned} \tag{5.A.6}$$

where in the last inequality we used the relative smoothness assumption in (5.A.1).

Summing the inequalities (5.A.2), (5.A.4) and (5.A.6), we have

$$\begin{aligned} B_{t+1}D(x,v_{t+1}) - B_t D(x,v_t) &\leq a_{t+1}f(x) - A_{t+1}f(x_{t+1}) + A_t f(x_t) + a_{t+1}(\psi(v_{t+1}) - \psi(x)) \\ &\leq -A_{t+1}\Big(f(x_{t+1}) - f(x)\Big) + A_t\Big(f(x_t) - f(x)\Big) + a_{t+1}\Big(\psi(x) - \psi(v_{t+1})\Big), \end{aligned}$$

which is the same as

$$\begin{aligned} &A_{t+1}\Big(f(x_{t+1}) - f(x)\Big) + B_{t+1}D(x,v_{t+1}) \leq \\ &\qquad A_t\Big(f(x_t) - f(x)\Big) + B_t D(x,v_t) + a_{t+1}\Big(\psi(x) - \psi(v_{t+1})\Big). \end{aligned} \tag{5.A.7}$$

Finally we consider the term $a_{t+1}\Big(\psi(x)-\psi(v_{t+1})\Big)$. Using $x_{t+1}=(1-\alpha_t)x_t+\alpha_t v_{t+1}$ and convexity of ψ, we have

$$\psi(x_{t+1})\le(1-\alpha_t)\psi(x_t)+\alpha_t\psi(v_{t+1}).$$

Since by definition $\alpha_t=\frac{a_{t+1}}{A_{t+1}}$ and $(1-\alpha_t)=\frac{A_t}{A_{t+1}}$, the above inequality is equivalent to

$$A_{t+1}\psi(x_{t+1})\le A_t\psi(x_t)+a_{t+1}\psi(v_{t+1}),$$

which implies (using $A_{t+1}=A_t+a_{t+1}$) that for any $x\in\mathrm{dom}\phi$,

$$A_{t+1}\Big(\psi(x_{t+1})-\psi(x)\Big)\le A_t\Big(\psi(x_t)-\psi(x)\Big)+a_{t+1}\Big(\psi(v_{t+1})-\psi(x)\Big). \tag{5.A.8}$$

Summing the inequalities (5.A.7) and (5.A.8) and using $\Phi=f+\psi$, we have

$$A_{t+1}\Big(\Phi(x_{t+1})-\Phi(x)\Big)+B_{t+1}D(x,v_{t+1})\le A_t\Big(\Phi(x_t)-\Phi(x)\Big)+B_tD(x,v_t).$$

This can then be unrolled, and we obtain the desired result by setting $x=x_*$.

Finally, the estimates of A_t and B_t follow from a direct adaptation of the techniques in [Nesterov and Stich, 2017]. The only difference is the use of the time-varying parameter $\gamma_t=\sqrt{\sigma_{F/\phi}/(L_{F/\phi}G_t)}$ instead of a constant $\gamma=\sqrt{\sigma_{F/\phi}/L_{F/\phi}}$, which does not impact the derivations. □

5.A.3. Proof of Lemma 30. The analysis in Lemma 29 is very pessimistic, since we use uniform lower and upper bounds for the Hessian of ϕ, whereas what we actually want is to bound is the differences between Hessians. If the Hessian is well-behaved (typically Lipschitz, or if ϕ is self-concordant), we can prove Lemma 30, which leads to a finer asymptotic convergence rate.

We start with the local quadratic representation of Bregman divergence:

$$\begin{aligned}
D_\phi(x_{t+1},y_t)&=\frac12\|x_{t+1}-y_t\|^2_{H(x_{t+1},y_t)}=\frac{\alpha_t^2}{2}\|v_{t+1}-w_t\|^2_{H(x_{t+1},y_t)}\\
&\le\frac{\alpha_t^2}{2}\Big((1-\beta_t)\|v_{t+1}-v_t\|^2_{H(x_{t+1},y_t)}+\beta_t\|v_{t+1}-y_t\|^2_{H(x_{t+1},y_t)}\Big)\\
&\le\frac{\alpha_t^2}{2}\Big((1-\beta_t)\|v_{t+1}-v_t\|^2_{H(v_{t+1},v_t)}+\beta_t\|v_{t+1}-y_t\|^2_{H(v_{t+1},y_t)}\Big)\\
&\quad+\frac{\alpha_t^2}{2}(1-\beta_t)\|H(x_{t+1},y_t)-H(v_{t+1},v_t)\|\cdot\|v_{t+1}-v_t\|^2\\
&\quad+\frac{\alpha_t^2}{2}\beta_t\|H(x_{t+1},y_t)-H(v_{t+1},y_t)\|\cdot\|v_{t+1}-y_t\|^2.
\end{aligned}$$

Now we use the Lipschitz property of $\nabla^2\phi$ to bound the spectral norms of differences of Hessians:

$$\|H(x_{t+1},y_t)-H(v_{t+1},v_t)\|\le M\|z_{xy}-z_{vv}\|,\qquad\|H(x_{t+1},y_t)-H(v_{t+1},y_t)\|\le M\|z_{xy}-z_{vy}\|,$$

where $z_{vv}\in[v_{t+1},v_t]$, $z_{xy}\in[y_t,x_{t+1}]$ and $z_{vy}\in[y_t,v_{t+1}]$. Using the triangle inequality of norms, we have

$$\|z_{xy}-z_{vy}\|=\|z_{xy}-y_t+y_t-z_{vy}\|\le\|z_{xy}-y_t\|+\|y_t-z_{vy}\|\le\|x_{t+1}-y_t\|+\|y_t-v_{t+1}\|,$$

and

$$\|z_{vv} - z_{xy}\| \leq \|z_{vv} - v_{t+1}\| + \|v_{t+1} - y_t\| + \|y_t - z_{xy}\| \leq \|v_t - v_{t+1}\| + \|v_{t+1} - y_t\| + \|y_t - x_{t+1}\|.$$

Therefore, we have

$$d_t \triangleq \max\Big\{\|z_{xy} - z_{vv}\|,\ \|z_{xy} - z_{vy}\|\Big\} \leq \|v_t - v_{t+1}\| + \|v_{t+1} - y_t\| + \|y_t - x_{t+1}\|,$$

and consequently,

$$\begin{aligned}
D_\phi(x_{t+1}, y_t) &\leq \frac{\alpha_t^2}{2}\Big((1-\beta_t)\|v_{t+1} - v_t\|^2_{H(v_{t+1},v_t)} + \beta_t\|v_{t+1} - y_t\|^2_{H(v_{t+1},y_t)}\Big) \\
&\quad + \frac{Md_t\alpha_t^2}{2}\Big((1-\beta_t)\|v_{t+1} - v_t\|^2 + \beta_t\|v_{t+1} - y_t\|^2\Big) \\
&\leq \frac{\alpha_t^2}{2}\Big((1-\beta_t)\|v_{t+1} - v_t\|^2_{H(v_{t+1},v_t)} + \beta_t\|v_{t+1} - y_t\|^2_{H(v_{t+1},y_t)}\Big) \\
&\quad + \frac{Md_t\alpha_t^2}{2\sigma_\phi}\Big((1-\beta_t)\|v_{t+1} - v_t\|^2_{H(v_{t+1},v_t)} + \beta_t\|v_{t+1} - y_t\|^2_{H(v_{t+1},y_t)}\Big) \\
&= \frac{\alpha_t^2}{2}\left(1 + \frac{Md_t}{\sigma_\phi}\right)\Big((1-\beta_t)\|v_{t+1} - v_t\|^2_{H(v_{t+1},v_t)} + \beta_t\|v_{t+1} - y_t\|^2_{H(v_{t+1},y_t)}\Big) \\
&= \alpha_t^2\left(1 + \frac{Md_t}{\sigma_\phi}\right)\Big((1-\beta_t)D(v_{t+1}, v_t) + \beta_t D(v_{t+1}, y_t)\Big).
\end{aligned}$$

Combining with Lemma 29, we see that $G_t = \min\{\kappa_\sigma,\ 1 + (M/\sigma_\phi)d_t\}$ satisfies the inequality (5.3.2). This finishes the proof of Lemma 30.

Note that this condition is not directly useful. Indeed, x_{t+1} and v_{t+1} depend on G_t. Yet, under the uniform choice of $G_t \leq \kappa_\phi$, it can be shown that $d_t \to 0$ at rate $(1 - 1/\sqrt{\kappa_\phi\kappa_{\text{rel}}})^t$ because the sequences v_t, x_t and y_t all converge to x^* at this rate in the strongly convex case [Lin and Xiao, 2015, Theorem 1]. As a consequence, Algorithm 10 will eventually use $G_t \leq 2$, leading to an asymptotic rate of $(1 - 1/\sqrt{\kappa_{\text{rel}}})^t$.

5.B. Concentration of Hessians

In practice, preconditioned gradient methods such as DANE are often used with a stepsize of 1. This implies the assumption of $L_{\text{rel}} = 1$, which holds if n is sufficiently large with a given μ or if μ is sufficiently large for a given n (but $\mu \leq L_F$ always). Otherwise convergence is not guaranteed (which is why it is sometimes considered as "rather unstable"). If μ is such that $\|H_f(x) - H_F(x)\| \leq \mu$ for all $x \in \mathcal{B}(0, D)$ then $L_{\text{rel}} = 1$ can safely be chosen since $H_F(x) - H_f(x) \preceq \mu I_d$. Note that this choice of μ is completely independent of λ. In this case, we use that $H_F(x) - H_f(x) \succeq -\mu I$ to write that

$$H_f(x) + \mu \preceq H_F(x) + 2\mu \preceq (1 + 2\mu H_F^{-1}(x))H_F(x) \preceq \left(1 + \frac{2\mu}{\lambda}\right)H_F(x).$$

These derivations are similar to the ones of Zhang and Xiao [2018, Lemma 3], and so we obtain $\sigma_{\text{rel}} = \left(1 + \frac{2\mu}{\lambda}\right)^{-1}$ and the corresponding relative condition number $\kappa_{\text{rel}} = 1 + \frac{2\mu}{\lambda}$, as explained in Section 5.1.1. We see that μ is independent of λ, but the problem is still very ill-conditioned for small values of λ, meaning that acceleration makes a lot of sense. In the quadratic case, tighter relative bounds can be derived.

5.B.1. The quadratic case. This section is focusing on proving Theorem 27.

PROOF OF THEOREM 27. We consider the random variable a, and $(a_i)_{i\in\{1,\dots,n\}}$ are n i.i.d. variables with the same law as a. We introduce matrices $\hat{H}$ and H such that $H_f = \hat{H} + \lambda I_d$ and $H_F = H + \lambda I_d$. In particular, $H = \mathbb{E}[aa^\top] = \mathbb{E}\hat{H}$. We define for $\alpha \geq 0$, $\beta > 0$, $H_{\alpha,\beta} = \alpha H + \beta I_d$, and

$$S_i = \frac{1}{n} H_{\alpha,\beta}^{-\frac{1}{2}} (a_i a_i^\top - H) H_{\alpha,\beta}^{-\frac{1}{2}},$$

which is such that $\mathbb{E}[S_i] = 0$. This allows to have bounds of the form $\|\sum_i S_i\| \leq t$ with probability $1-\delta$ and a spectral bound μ that depends on α, β, δ (and other quantities related to H and $a_i a_i^\top$). We note that

$$\sum_{i=1}^n S_i = H_{\alpha,\beta}^{-\frac{1}{2}} (\hat{H} - H) H_{\alpha,\beta}^{-\frac{1}{2}},$$

and write the concentration bounds on the S_i as $-tH_{\alpha,\beta} \preceq \hat{H} - H \preceq tH_{\alpha,\beta}$ for some $t > 0$, which can be rearranged as:

$$\hat{H} + t\beta I_d \succeq (1 - t\alpha) H$$
$$\hat{H} - t\beta I_d \preceq (1 + t\alpha) H.$$

Using $H_f = \hat{H} + \lambda I_d$ and $H_F = H + \lambda I_d$, the first equation can be rearranged as:

$$H_F \preceq \frac{1}{1 - t\alpha} \left(H_f + t(\beta - \alpha\lambda) I_d \right). \tag{5.B.1}$$

The second equation can be written

$$H_f \preceq \left[(1 + t\alpha) I_d + t(\beta - \alpha\lambda) H_F^{-1} \right] H_F,$$

which, by adding $t(\beta - \alpha\lambda) I_d$ on both sides, leads to

$$H_f + t(\beta - \alpha\lambda) I_d \preceq \left[(1 + t\alpha) I_d + 2t(\beta - \alpha\lambda) H_F^{-1} \right] H_F.$$

We let $\mu = t(\beta - \alpha\lambda)$ and use $H_F^{-1} \preceq \lambda^{-1} I_d$ to write that:

$$\left(1 + \alpha t + \frac{2\mu}{\lambda} \right)^{-1} (H_f + \mu I_d) \preceq H_F \preceq \frac{1}{1 - \alpha t} (H_f + \mu I_d). \tag{5.B.2}$$

We then use the fact that $a_i a_i^\top$ and H are positive semidefinite and upper bounded by $R^2 I$ to write that:

$$\|S_i\| \leq \frac{1}{n} \|H_{\alpha,\beta}^{-1}\| \max\left\{ \|aa^\top\|, \|H\| \right\} \leq \frac{R^2}{\beta n}. \tag{5.B.3}$$

Using the fact that $H = \mathbb{E}[aa^\top]$, we bound the variance as:

$$\begin{aligned}
\left\| \sum_i \mathbb{E}[S_i S_i^\top] \right\| &= \frac{1}{n} \left\| \mathbb{E}[H_{\alpha,\beta}^{-\frac{1}{2}} (aa^\top - H) H_{\alpha,\beta}^{-1} (aa^\top - H) H_{\alpha,\beta}^{-\frac{1}{2}}] \right\| \\
&= \frac{1}{n} \left\| H_{\alpha,\beta}^{-\frac{1}{2}} (\mathbb{E}[aa^\top H_{\alpha,\beta}^{-1} aa^\top] - H H_{\alpha,\beta}^{-1} H) H_{\alpha,\beta}^{-\frac{1}{2}} \right\| \\
&\leq \frac{1}{n} \max\left\{ \tilde{R}^2 \left\| H_{\alpha,\beta}^{-\frac{1}{2}} \mathbb{E}[aa^\top] H_{\alpha,\beta}^{-\frac{1}{2}} \right\|, \left\| H_{\alpha,\beta}^{-\frac{1}{2}} H H_{\alpha,\beta}^{-1} H H_{\alpha,\beta}^{-\frac{1}{2}} \right\| \right\}
\end{aligned}$$

$$\leq \frac{1}{n}\left\|H_{\alpha,\beta}^{-\frac{1}{2}} H H_{\alpha,\beta}^{-\frac{1}{2}}\right\| \max\left\{\tilde{R}^2,\ \left\|H_{\alpha,\beta}^{-\frac{1}{2}} H H_{\alpha,\beta}^{-\frac{1}{2}}\right\|\right\},$$

with $\tilde{R}^2 \geq a^\top H_{\alpha,\beta}^{-1} a$ almost surely. We first notice that $a_i^\top H_{\alpha,\beta}^{-1} a_i \leq \frac{R^2}{\beta}$. Then, we use the positive definiteness of $H_{\alpha,\beta}$ and H and the fact that $\beta H_{\alpha,\beta}^{-1} \preceq I_d$ to show that for $\alpha > 0$:

$$\left\|H_{\alpha,\beta}^{-\frac{1}{2}} H H_{\alpha,\beta}^{-\frac{1}{2}}\right\| = \left\|H_{\alpha,\beta}^{-\frac{1}{2}} \frac{\alpha H + \beta - \beta}{\alpha} H_{\alpha,\beta}^{-\frac{1}{2}}\right\| = \frac{1}{\alpha}\left\|I_d - \beta H_{\alpha,\beta}^{-1}\right\| \leq \frac{1}{\alpha}\left(1 - \frac{\beta}{\alpha L + \beta}\right) = \frac{L}{\alpha L + \beta},$$

where L is the spectral norm of H, i.e., $L = \|H\|$. A quick calculation shows that this formula is also true for $\alpha = 0$. In the case $\alpha = 0$ and $\beta = 1$ (absolute bounds), $H_{\alpha,\beta} = I_d$ and we recover that we can bound the variance by $\frac{LR^2}{n}$, leading to the usual additive bounds.

For $\alpha > 0$, we use the simpler bound $\left\|H_{\alpha,\beta}^{-\frac{1}{2}} H H_{\alpha,\beta}^{-\frac{1}{2}}\right\| \leq \alpha^{-1}$ and $\tilde{R}^2 \leq \beta^{-1} R^2$, leading to

$$\left\|\sum_i \mathbb{E}[S_i S_i^\top]\right\| \leq \frac{\max(\beta^{-1}R^2, \alpha^{-1})}{n\alpha}.$$

For any $1 > \delta > 0$, we note $c_\delta = \frac{28}{3}\log\left(\frac{2d}{\delta}\right)$. We now set $\alpha = \frac{\beta n}{c_\delta R^2}$, and assume that $n > c_\delta$ (otherwise concentration bounds will be very loose anyway). In this case, $\beta^{-1}R^2 \geq \alpha^{-1}$, meaning that the bound on the variance becomes:

$$\left\|\sum_i \mathbb{E}[S_i S_i^\top]\right\| \leq \frac{1}{\alpha^2 c_\delta}.$$

Similarly, according to (5.B.3), every S_i is almost surely bounded as: $\|S_i\| \leq \frac{1}{\alpha c_\delta}$. We can now use Matrix Bernstein Inequality [Tropp, 2015, Theorem (6.1.1)] to get that with probability $1 - p_\delta$ and for $t \geq 0$,

$$\left\|\sum_{i=1}^n S_i\right\| \leq t,$$

with

$$p_\delta = 2d \cdot \exp\left(-\frac{t^2/2}{(\alpha^2 c_\delta)^{-1} + (\alpha c_\delta)^{-1} t/3}\right).$$

We choose $t = (2\alpha)^{-1}$, which leads to $p_\delta = \delta$. By substituting the expressions of $\alpha t = \frac{1}{2}$ and $\beta t = \frac{R^2 c_\delta}{n}\alpha t$ into Equation (5.B.2), we obtain:

$$\left(\frac{3}{2} + \frac{2\mu}{\lambda}\right)^{-1}\left(\hat{H}_\lambda + \mu I_d\right) \preceq H_\lambda \preceq 2\left(\hat{H}_\lambda + \mu I_d\right),$$

with

$$\mu = t(\beta - \alpha\lambda) = \frac{1}{2}\left(\frac{28R^2}{3n}\log\left(\frac{2d}{\delta}\right) - \lambda\right).$$

In case β is very small so that $\mu < 0$ then it is always possible to choose $\delta' < \delta$ so that $\mu > 0$. This means that the same bound on μ holds with probability $1 - \delta' > 1 - \delta$. □

5.B.2. Almost surely bounded a. We first introduce Theorem 32, which proves a general concentration result that implies Theorem 28 as a special case.

THEOREM 32. *We consider functions φ_1, φ_2, which are respectively L_1 and L_2 Lipschitz-continuous. We consider two sets $\mathcal{X}$ and $\mathcal{Y}$ which are contained in balls of center 0 and radius D_1 and D_2. We assume that $|\varphi_1(a_i^\top x)| \leq B_1$ and $|\varphi_2(a_i^\top y)| \leq B_2$ almost surely for all $x \in \mathcal{X}$ and $y \in \mathcal{Y}$. We consider*

$$Y = \sup_{x\in\mathcal{X},\ y\in\mathcal{Y}} \left\{ \frac{1}{n}\sum_{i=1}^n \varphi_1(a_i^\top x)\varphi_2(a_i^\top y) - \mathbb{E}\varphi_1(a^\top x)\varphi_2(a^\top y) \right\}.$$

Then, for all $1 \geq \delta > 0$, with probability greater than $1-\delta$:

$$Y \leq \sqrt{4\pi}\frac{(\mathbb{E}[\|a\|^2])^{\frac{1}{2}}}{\sqrt{n}}(B_2L_1D_1 + B_1L_2D_2) + \frac{2B_1B_2}{\sqrt{2n}}\sqrt{\log\frac{1}{\delta}}.$$

Theorem 28 is then a direct corollary of Theorem 32, as shown below:

PROOF OF THEOREM 28. The result is obtained by applying Theorem 32 with $\varphi_1 = \ell''$ and $\varphi_2 = \frac{1}{2}(\cdot)^2$. This implies that with probability at least $1-\delta$,

$$\sup_{x\in\mathcal{B}(0,D),\ y\in\mathcal{B}(0,1)} y^\top \left[\frac{1}{n}\sum_{i=1}^n \ell''(a_i^\top x) a_i a_i^\top - \mathbb{E}\ell''(a^\top x) aa^\top \right] y \leq \mu,$$

where the value of μ can be obtained by letting $B_1 = B_\ell$, $L_1 = M_\ell$, $D_1 = D$, $D_2 = 1$, $B_2 = \sup_{y:\|y\|\leq 1} y^\top a_i a_i^\top y \leq R^2$ and $L_2 = \sup_{y:\|y\|\leq 1} 2\|y^\top a_i\| = 2R$. □

PROOF OF THEOREM 32. If changing any a_i to some a_i', then the deviation in Y is at most (almost surely):

$$\frac{1}{n}\sup_{x\in\mathcal{X},\ y\in\mathcal{Y}} \left|\varphi_1(a_i^\top x)\varphi_2(a_i^\top y)\right| + \sup_{x\in\mathcal{X},\ y\in\mathcal{Y}} \left|\varphi_1(a_i'^\top x)\varphi_2(a_i'^\top y)\right| \leq \frac{2}{n}B_1B_2.$$

Mac-Diarmid's inequality [see, e.g., Vershynin, 2019, Theorem 2.9.1] thus implies that with probability greater than $1-\delta$,

$$Y \leq \mathbb{E}Y + \frac{2B_1B_2}{\sqrt{2n}}\sqrt{\log\frac{1}{\delta}}\ . \tag{5.B.4}$$

In order to bound $\mathbb{E}Y$, we first use classical symmetrization property [see, e.g., Vershynin, 2019, Section 6.4]

$$\mathbb{E}Y \leq \sqrt{2\pi}\cdot\mathbb{E}\sup_{x\in\mathcal{X},\ y\in\mathcal{Y}} \frac{1}{n}\sum_{i=1}^n \varepsilon_i\varphi_1(a_i^\top x)\varphi_2(a_i^\top y),$$

where each ε_i is an independent standard normal variable.

Denoting $Z_{x,y} = \frac{1}{n}\sum_{i=1}^n \varepsilon_i\varphi_1(a_i^\top x)\varphi_2(a_i^\top y)$, we have, for any x, y, x', y', assuming the a_i are fixed,

$$\begin{aligned}
\mathbb{E}(Z_{x,y} - Z_{x',y'})^2 &= \frac{1}{n^2}\sum_{i=1}^n \Big(\varphi_1(a_i^\top x)\varphi_2(a_i^\top y) - \varphi_1(a_i^\top x')\varphi_2(a_i^\top y')\Big)^2 \\
&= \frac{1}{n^2}\sum_{i=1}^n \Big(\varphi_1(a_i^\top x)\big[\varphi_2(a_i^\top y) - \varphi_2(a_i^\top y')\big] + \big[\varphi_1(a_i^\top x) - \varphi_1(a_i^\top x')\big]\varphi_2(a_i^\top y')\Big)^2
\end{aligned}$$

$$\leq \frac{1}{n^2}\sum_{i=1}^{n}\left(2\,\varphi_1(a_i^\top x)^2\Big[\varphi_2(a_i^\top y)-\varphi_2(a_i^\top y')\Big]^2+2\,\varphi_2(a_i^\top y')^2\Big[\varphi_1(a_i^\top x)-\varphi_1(a_i^\top x')\Big]^2\right)$$
$$\leq \frac{1}{n^2}\sum_{i=1}^{n}\left(2B_1^2\Big[\varphi_2(a_i^\top y)-\varphi_2(a_i^\top y')\Big]^2+2B_2^2\Big[\varphi_1(a_i^\top x)-\varphi_1(a_i^\top x')\Big]^2\right).$$

We then have, using Lipschitz-continuity:

$$\begin{aligned}\mathbb{E}(Z_{x,y}-Z_{x',y'})^2 &\leq \frac{1}{n^2}\sum_{i=1}^{n}\left(2B_1^2L_2^2\Big[a_i^\top y-a_i^\top y'\Big]^2+2B_2^2L_1^2\Big[a_i^\top x-a_i^\top x'\Big]^2\right)\\ &= \mathbb{E}(\tilde{Z}_{x,y}-\tilde{Z}_{x',y'})^2,\end{aligned}$$

for

$$\tilde{Z}_{x,y}=\frac{1}{n}\sum_{i=1}^{n}\left\{\sqrt{2}B_2L_1\tilde{\varepsilon}_{1i}a_i^\top x+\sqrt{2}B_1L_2\tilde{\varepsilon}_{2i}a_i^\top y\right\},$$

with all $\tilde{\varepsilon}_{1i}$ and $\tilde{\varepsilon}_{2i}$ independent standard random variables.

Using Sudakov-Fernique inequality [Vershynin, 2019, Theorem 7.2.11], we get

$$\begin{aligned}\mathbb{E}Y &= \sqrt{2\pi}\mathbb{E}\sup_{x\in\mathcal{X},\ y\in\mathcal{Y}}Z_{x,y}\\ &\leq \sqrt{2\pi}\mathbb{E}\sup_{x\in\mathcal{X},\ y\in\mathcal{Y}}\tilde{Z}_{x,y}\\ &= \sqrt{4\pi}B_2L_1\mathbb{E}\sup_{x\in\mathcal{X}}\frac{1}{n}\sum_{i=1}^{n}\tilde{\varepsilon}_{1i}a_i^\top x+\sqrt{4\pi}B_1L_2\mathbb{E}\sup_{y\in\mathcal{Y}}\frac{1}{n}\sum_{i=1}^{n}\tilde{\varepsilon}_{2i}a_i^\top y\\ &\leq \sqrt{4\pi}B_2L_1D_1\mathbb{E}\left\|\frac{1}{n}\sum_{i=1}^{n}\tilde{\varepsilon}_{1i}a_i\right\|+\sqrt{4\pi}B_1L_2D_2\mathbb{E}\left\|\frac{1}{n}\sum_{i=1}^{n}\tilde{\varepsilon}_{2i}a_i\right\|\\ &\leq \sqrt{4\pi}B_2L_1D_1\sqrt{\mathbb{E}\left\|\frac{1}{n}\sum_{i=1}^{n}\tilde{\varepsilon}_{1i}a_i\right\|^2}+\sqrt{4\pi}B_1L_2D_2\sqrt{\mathbb{E}\left\|\frac{1}{n}\sum_{i=1}^{n}\tilde{\varepsilon}_{2i}a_i\right\|^2}\\ &\leq \sqrt{4\pi}B_2L_1D_1\frac{(\mathbb{E}[\|a\|^2])^{\frac{1}{2}}}{\sqrt{n}}+\sqrt{4\pi}B_1L_2D_2\frac{(\mathbb{E}[\|a\|^2])^{\frac{1}{2}}}{\sqrt{n}}.\end{aligned}$$

Plugging this into Equation (5.B.4), we obtain that with probability greater than $1-\delta$,

$$Y\leq\sqrt{4\pi}\frac{(\mathbb{E}[\|a\|^2])^{\frac{1}{2}}}{\sqrt{n}}(B_2L_1D_1+B_1L_2D_2)+\frac{2B_1B_2}{\sqrt{2n}}\sqrt{\log\frac{1}{\delta}}.$$

□

Remark 3 (Relative bounds). In the quadratic case, considering relative bounds allowed to choose smaller values of μ and to tighten the bounds on the relative condition number by a $\sqrt{n}$ factor. Theorem 28 consists in bounding (using the definition of the operator norm)

$$\sup_{x\in\mathcal{B}(0,D),y\in\mathcal{B}(0,1)}\left\{\frac{1}{n}\sum_{i=1}^{n}\ell''(a_i^\top x)(a_i^\top y)^2-y^\top H(x)y\right\},$$

and heavily relies on the fact that $(a_i^\top y)^2$ is independent of x. The proof needs to be adapted in the case of the relative bounds since this term becomes $(a_i^\top H_{\alpha,\beta}^{-\frac{1}{2}}(x)y)^2$, which now depends on x as well, and thus requires a different control.

5.B.3. Subgaussian a. We considered in the previous section a splitting of the summands of the Hessians as a product of 2 functions. We now present a different bound that is designed for a product of an arbitrary number of functions $\varphi_1, \dots, \varphi_r : \mathbb{R} \to \mathbb{R}$. This section is devoted to proving Theorem 33, which is based on the chaining argument [Boucheron et al., 2013, Chapter 13], and from which Theorem 29 can be derived directly.

THEOREM 33. *Assume that for all i, $\varphi_i(0) = 0$ and φ_i is 1-Lipschitz. Assume that a is ρ-subgaussian, and that for all k, $\sup_{x \in \mathcal{B}(0,1)} |\varphi_k(a_i^\top x)| \le B_k$. Denote $B = \prod_{k=1}^r B_k$. For suitable constant C_r, for all $\gamma > 0$, one has that*

$$\mathbb{P}\left(\sup_{x_1,\dots,x_r \in \mathcal{B}(0,1)} \frac{1}{n} \sum_{i \in [n]} \left\{ \prod_{k=1}^r \varphi_k(a_i^\top x_k) - \mathbb{E} \prod_{k=1}^r \varphi_k(a^\top x_k) \right\} \ge \rho^r C_r (d+\gamma) \left[\frac{1}{\sqrt{dn}} + \frac{(\rho^{-r} B)^{1-2/r}}{n} \right] \right)$$
$$\le r \frac{\pi^2}{6} e^{-\gamma}.$$

We are primarily interested in the case $r = 3$, $\varphi_1 = \varphi_2 = \mathrm{id}$ (the identity mapping) to control distances between Hessians.

PROOF. We look for bounds on

$$Y := \sup_{x_1,\dots,x_r \in \mathcal{S}_1} \frac{1}{n} \sum_{i \in [n]} \left\{ \prod_{k=1}^r \varphi_k(a_i^\top x_k) - \mathbb{E}_a \prod_{k=1}^r \varphi_k(a^\top x_k) \right\}. \tag{5.B.5}$$

For all $j \ge 0$, let $\mathcal{N}_j$ be an ϵ-net of $\mathcal{S}_1$ that approximates $\mathcal{S}_1$ to distance 2^{-j}. Then, $\mathcal{N}_j$ can be chosen as $|\mathcal{N}_j| \le (1+2^{j+1})^d$[see, e.g., Vershynin, 2019, Section 4.2]. For all $x \in \mathcal{S}_1$, let $\Pi_j(x)$ be some point in $\mathcal{N}_j$ such that $\|x - \Pi_j(x)\| \le 2^{-j}$. By convention we take $\Pi_0(x) = 0$.

Then for all $(x_1, \dots, x_r) \in \mathcal{S}^r$, using the chaining approach [Boucheron et al., 2013], we write

$$\frac{1}{n} \sum_{i \in [n]} \prod_{k \in [r]} \varphi_k(a_i^\top x_k)$$
$$= \sum_{j \ge 0} \frac{1}{n} \sum_{i \in [n]} \left\{ \prod_{k \in [r]} \varphi_k(a_i^\top \Pi_{j+1}(x_k)) - \prod_{k \in [r]} \varphi_k(a_i^\top \Pi_j(x_k)) \right\}$$
$$= \sum_{j \ge 0} \sum_{k \in [r]} \frac{1}{n} \sum_{i \in [n]} \prod_{\ell=1}^{k-1} \varphi_\ell(a_i^\top \Pi_{j+1}(x_\ell)) \Big[\varphi_k(a_i^\top \Pi_{j+1}(x_k)) - \varphi_k(a_i^\top \Pi_j(x_k)) \Big] \prod_{\ell=k+1}^{r} \varphi_\ell(a_i^\top \Pi_j(x_\ell)).$$

Let $j \ge 0$ and $k \in [r]$ be fixed. Consider a term of the form $Z = \frac{1}{n} \sum_{i \in [n]} Z_i$, with

$$Z_i = \prod_{\ell=1}^{k-1} \varphi_\ell(a_i^\top u_\ell) \Big[\varphi_k(a_i^\top u_k) - \varphi_k(a_i^\top v_k) \Big] \prod_{\ell=k+1}^{r} \varphi_\ell(a_i^\top v_\ell), \tag{5.B.6}$$

where $u_\ell \in \mathcal{N}_j$, $v_\ell \in \mathcal{N}_{j+1}$, and $\|u_k - v_k\| \le \epsilon_j := 2^{-j+1}$. By the triangle inequality, for all $x_\ell \in \mathcal{S}_1$, letting $u_\ell = \Pi_j(x_\ell)$ and $v_\ell = \Pi_{j+1}(x_\ell)$, these assumptions are satisfied. For each Z_i and $t > 0$, we have:

$$\mathbb{P}(Z_i \ge \epsilon_j \rho^r t) \le \mathbb{P}\Big(|\varphi_\ell(a_i^\top u_\ell)| \ge \rho t^{1/r} \text{ for some } \ell < r,$$
$$\text{or } |\varphi_k(a_i^\top u_k) - \varphi_k(a_i^\top v_k)| \ge \rho \epsilon_j t^{1/r},$$

$$\text{or } |\varphi_\ell(a_i^\top v_\ell)| \geq \rho t^{1/r} \text{ for some } \ell > k\Big).$$

Therefore, we have

$$\mathbb{P}(Z_i \geq \epsilon_j \rho^r t) \leq 2re^{-t^{2/r}/2} \text{ if } t \leq \rho^{-r} P_{j,k} \text{ and}$$
$$\mathbb{P}(Z_i \geq \epsilon_j \rho^r t) = 0 \text{ if } t > \rho^{-r} P_{j,k},$$

where we noted $P_{j,k} := \min\Big\{2B/\epsilon_j, 2(B/B_k)R\Big\}$. We will also make use of notation $j^*(k) := \lceil \log_2(R/B_k) \rceil$, so that

$$j \leq j^*(k) \Rightarrow P_{j,k} = 2B/\epsilon_j, \qquad j > j^*(k) \Rightarrow P_{j,k} = 2(B/B_k)R.$$

Fixing $j \geq 0$, $k \in [r]$, we write for any $\theta > 0$ (a specific θ will be chosen later):

$$\mathbb{E}\, e^{(\theta/n)\rho^{-r}[Z_i - \mathbb{E}Z_i]/\epsilon_j} = 1 + \left(\frac{\theta}{n}\right)^2 \mathbb{E}\Big[(\epsilon_j^{-1}\rho^{-r}(Z_i - \mathbb{E}Z_i))^2 F((\theta/n)(\epsilon_j^{-1}\rho^{-r}(Z_i - \mathbb{E}Z_i)))\Big],$$

where

$$F(x) := x^{-2}[e^x - x - 1] \leq e^{|x|}.$$

Thus using this bound and the inequality $xy \leq x^2 + y^2$:

$$\mathbb{E}\, e^{(\theta/n)\rho^{-r}[Z_i - \mathbb{E}Z_i]/\epsilon_j} \leq 1 + \left(\frac{\theta}{n}\right)^2 \Big[\mathbb{E}((\epsilon_j^{-1}\rho^{-r}(Z_i - \mathbb{E}Z_i))^4 + \mathbb{E}e^{2(\theta/n)\rho^{-r}|Z_i - \mathbb{E}Z_i|/\epsilon_j}\Big]. \quad (5.B.7)$$

By the sub-gaussian tail assumption, $\mathbb{E}(\epsilon_j^{-1}\rho^{-r}(Z_i - \mathbb{E}Z_i))^4$ is bounded by a constant κ_r dependent on r. We now assume that θ is such that

$$\frac{\theta}{n} \leq \min\left(\frac{(\rho^{-r}P_{j,k})^{2/r-1}}{8}, 1\right),$$

which is equivalent to having $(\theta/n)y \leq y^{2/r}/8$ for $y \in [0, \rho^{-r}P_{j,k}]$ and $r \geq 2$. Then, $\mathbb{E}e^{2(\theta/n)\rho^{-r}|Z_i - \mathbb{E}Z_i|/\epsilon_j}$ is also bounded by another constant κ_r' dependent on r. Indeed, by the sub-gaussian tail assumption, $|\mathbb{E}Z_i| \leq \rho^r \epsilon_j s_r$ for some r-dependent constant, and we can then use the fact that:

$$\begin{aligned}
\mathbb{E}e^{\alpha X} &= \int_0^\infty e^{kz} p(z) dz \\
&= \int_0^\infty \left(1 + \alpha \int_0^z e^{\alpha y}\right) p(z) dz dy \\
&= 1 + \alpha \int_0^\infty \int_y^\infty e^{\alpha y} dy p(z) dz \\
&= 1 + \alpha \int_0^\infty e^{\alpha y} p(X \geq y) dy,
\end{aligned}$$

with $\alpha = 2\theta/n$ and $X = \rho^{-r}|Z_i|\epsilon_j$ to get:

$$\begin{aligned}
\mathbb{E}\, e^{2(\theta/n)\rho^{-r}|Z_i - \mathbb{E}Z_i|/\epsilon_j} &\leq \mathbb{E}\, e^{2(\theta/n)\rho^{-r}(|Z_i| + |\mathbb{E}Z_i|)/\epsilon_j} \\
&\leq e^{2(\theta/n)s_r} \mathbb{E}e^{2\theta/n\rho^{-r}|Z_i|/\epsilon_j} \\
&\leq e^{2(\theta/n)s_r}[1 + \frac{2\theta}{n}\int_0^\infty e^{2(\theta/n)y}[\mathbb{P}(Z_i \geq y\rho^r\epsilon_j) + \mathbb{P}(-Z_i \geq y\rho^r\epsilon_j)]dy] \\
&\leq e^{2(\theta/n)s_r}[1 + \frac{2\theta}{n} 2r \int_0^{\rho^{-r}P_{j,k}} e^{2(\theta/n)y - y^{2/r}/2} dy],
\end{aligned}$$

$$\leq e^{2(\theta/n)s_r}[1+\frac{2\theta}{n}2r\int_0^\infty e^{-y^{2/r}/4}dy]$$
$$= e^{2(\theta/n)s_r}[1+\frac{\theta}{n}c_r].$$

We finally use the fact that $\theta/n \leq 1$ to write $\mathbb{E}e^{2(\theta/n)\rho^{-r}|Z_i-\mathbb{E}Z_i|/\epsilon_j} \leq \kappa_r'$, with $\kappa_r' = e^{2s_r}[1+c_r]$. We write $\kappa_r'' = \kappa_r + \kappa_r'$ and use Equation (5.B.7) together with the independence of the Z_i to obtain:

$$\mathbb{E}\, e^{\theta\rho^{-r}\left[\frac{1}{n}\sum_{i=1}^n Z_i-\mathbb{E}Z_i\right]/\epsilon_j} \leq \left(1+\left(\frac{\theta}{n}\right)^2\kappa_r''\right)^n \leq e^{\frac{\theta^2}{n}\kappa_r''}.$$

Thus, using that $\mathbb{P}(X \geq y) = \mathbb{P}(e^X \geq e^y) \leq e^{-y}\mathbb{E}e^X$ (Markov Inequality), we have that for fixed vertices $u_\ell, v_\ell,\ \ell \in [r]$ in the suitable ϵ-nets, this probability is upper bounded for all $\theta \in [0, \min(n, n(\rho^{-r}P_{j,k})^{2/r-1}/8)]$ as:

$$\mathbb{P}\left(\frac{1}{n}\sum_{i\in[n]} Z_i - \mathbb{E}Z_i \geq \rho^r\epsilon_j t_{j,k}\right) \leq \exp\left((r+1)d\ln(1+2^{j+2}) - \theta t_{j,k} + \kappa_r''\theta^2/n\right). \quad (5.B.8)$$

We see in Equation (5.B.6) that the variables Z_i are built by fixing a specific either u_ℓ for $\ell < k$, v_ℓ for $\ell > k$, and u_k and v_k, meaning that there are actually $r+1$ variables to be fixed in nets of resolution either 2^{-j} or 2^{-j-1}. Note that all Z_i for $i \in \{1, \cdots, n\}$ are constructed with the same choice of u_ℓ and v_ℓ. Therefore, the number of possible choices for $u_\ell \in \mathcal{N}_j$ and $v_\ell \in \mathcal{N}_{j+1}$ involved in the definition of Z_i is upper-bounded by

$$|\mathcal{N}_{j+1}|^{r+1} \leq e^{d(r+1)\ln(1+2^{j+2})}.$$

Combining this with Equation (5.B.8), we obtain using a union bound that:

$$\begin{aligned}\mathbb{P}\left(\sup_{u_\ell,v_\ell}\left\{Z-\mathbb{E}Z\right\} \geq \rho^r\epsilon_j t_{j,k}\right) &= \mathbb{P}\left(\cup_{u_\ell,v_\ell}\left\{Z-\mathbb{E}Z \geq \rho^r\epsilon_j t_{j,k}\right\}\right)\\ &\leq \sum_{u_\ell,v_\ell}\mathbb{P}\left(Z-\mathbb{E}Z \geq \rho^r\epsilon_j t_{j,k}\right)\\ &\leq \exp\left((r+1)d\ln(1+2^{j+2}) - \theta t_{j,k} + \kappa_r''\theta^2/n\right).\end{aligned}$$

Let now $\theta_{j,k} = \min(n, n(\rho^{-r}P_{j,k})^{2/r-1}/8, \sqrt{nd})$, and

$$t_{j,k} = \kappa_r''\frac{\theta_{j,k}}{n} + \frac{1}{\theta_{j,k}}[d(r+1)\ln(1+2^{j+2}) + \gamma + 2\ln(j+1)],$$

where $\gamma > 0$ is a free parameter. We then use the chaining decomposition of

$$Y = \sup_x\left\{\sum_{j\geq 0, k\in[r]} Z - \mathbb{E}Z\right\},$$

and another union bound on j and k to write that:

$$\mathbb{P}\left(Y \geq \rho^r\sum_{j\geq 0,k\in[r]}\epsilon_j t_{j,k}\right) = \mathbb{P}\left(\sup_x\left\{\sum_{j\geq 0,k\in[r]} Z-\mathbb{E}Z\right\} \geq \rho^r\sum_{j\geq 0,k\in[r]}\epsilon_j t_{j,k}\right)$$

$$
\begin{aligned}
&\leq \mathbb{P}\left(\sum_{j\geq 0,k\in[r]} \sup_x\{Z-\mathbb{E}Z\} \geq \rho^r \sum_{j\geq 0,k\in[r]} \epsilon_j t_{j,k}\right)\\
&\leq \sum_{j\geq 0,k\in[r]} \mathbb{P}\left(\sup_{u_\ell,v_\ell}\{Z-\mathbb{E}Z\} \geq \rho^r \epsilon_j t_{j,k}\right)\\
&\leq \sum_{j\geq 0,k\in[r]} e^{-\gamma-2\ln(1+j)}.
\end{aligned}
$$

In the end, using that $\sum_{j\geq 1} j^{-2} = \pi^2/6$, we obtain:

$$
\mathbb{P}\left(Y \geq \rho^r \sum_{j\geq 0,k\in[r]} \epsilon_j t_{j,k}\right) \leq r\frac{\pi^2}{6}e^{-\gamma}.
$$

Moreover, one has

$$
\epsilon_j t_{j,k} \leq \epsilon_j A_r(1+j)(d+\gamma)\left[\frac{(\rho^{-r}P_{j,k})^{1-2/r}}{n} + \frac{1}{\sqrt{nd}} + \frac{1}{n}\right],
$$

for some suitable constant A_r dependent only on r. Fix some $k \in [r]$. Write:

$$
\begin{aligned}
\frac{1}{A_r}\sum_{j\geq 0}\epsilon_j t_{j,k} &\leq 4\frac{d+\gamma}{\sqrt{nd}} + \frac{d+\gamma}{n}\sum_{j=0}^{j^*(k)} \epsilon_j(2\rho^{-r}B/\epsilon_j)^{1-2/r}(1+j)\\
&\quad + \frac{d+\gamma}{n}\sum_{j>j^*(k)} \epsilon_j(2\rho^{-r}BR/B_k)^{1-2/r}(1+j)\\
&\leq 4\frac{d+\gamma}{\sqrt{nd}} + \frac{d+\gamma}{n}A'_r(\rho^{-r}B)^{1-2/r}\left\{1+(B_k/R)^{2/r}\ln(R/B_k)\right\},
\end{aligned}
$$

where A'_r is another constant depending only on r. Since $B_k \leq R$, then $(B_k/R)^{2/r}\ln(R/B_k)$ is bounded by a (r-dependent) constant. □

We know present Corollary 10, which is a consequence of Theorem 33. We consider again i.i.d. a_i, bounded by R, satisfying the subgaussian tail assumption with parameter ρ, and some function φ that is 1-Lipschitz, and uniformly bounded by B_φ. Writing

$$
H(x) = \frac{1}{n}\sum_{i=1}^n a_i a_i^\top \varphi(a_i^\top x), \tag{5.B.9}
$$

We have the following corollary.

Corollary 10. *Thus for $1 > \delta > 0$, with probability at least $1-\delta$, it holds for some $C > 0$ that*

$$
\sup_{x\in\mathcal{S}_1}\|H(x)\|_{op} \leq C\rho^3(d+\ln(1/\delta)+\ln(5\pi^2/6))\left[\frac{1+\rho^{-1}B_\varphi}{\sqrt{dn}} + \frac{1+\{\rho^{-3}R^2B_\varphi\}^{1-2/3}}{n}\right]. \tag{5.B.10}
$$

Proof. Let us write $\varphi_3(u) = \varphi(u)-\varphi(0)$. Then φ_3 satisfies our assumptions (1-Lipschitz, $\varphi_3(0)=0$). Moreover, we can decompose matrix $H(x)-\mathbb{E}H(x)$ into $M(x)+N$, where

$$
M(x) = \frac{1}{n}\sum_{i=1}^n\left[a_i a_i^\top\varphi_3(a_i^\top x) - \mathbb{E}a_1a_1^\top\varphi_3(a_1^\top x)\right], \qquad N = \frac{1}{n}\sum_{i=1}^n\left[a_i a_i^\top\varphi(0) - \mathbb{E}a_1a_1^\top\varphi(0)\right].
$$

Taking $r = 2$, $\varphi_1 = \varphi_2 = Id$, the Theorem 33 gives us that

$$\mathbb{P}\left(\|N\|_{op} \geq C_2|\varphi(0)|\rho^2(\gamma + d)\left(1/\sqrt{dn} + 1/n\right)\right) \leq 2\frac{\pi^2}{6}e^{-\gamma}. \tag{5.B.11}$$

Taking next $r = 3$, and $B = R^2 B_\varphi$, we obtain

$$\mathbb{P}\left(\sup_{x\in\mathcal{S}_1}\|M(x)\|_{op} \geq C_3\rho^3(d + \gamma)\left(1/\sqrt{dn} + [\rho^{-3}R^2B_\varphi]^{1-2/3}/n\right)\right) \leq 3\frac{\pi^2}{6}e^{-\gamma}. \tag{5.B.12}$$

Combined, these two bounds give us that for all $\gamma > 0$, with $C = C_2 + C_3$:

$$\mathbb{P}\left(\sup_{x\in\mathcal{S}_1}\|H(x)\|_{op} \geq C\rho^3(d + \gamma)\left[\frac{1 + \rho^{-1}B_\varphi}{\sqrt{dn}} + \frac{1 + \{\rho^{-3}R^2B_\varphi\}^{1-2/3}}{n}\right]\right) \leq 5\frac{\pi^2}{6}e^{-\gamma}. \tag{5.B.13}$$

We finally take $\gamma = -\ln\left(\frac{6\delta}{5\pi^2}\right)$. □

The last step required to prove Theorem 29 is to consider the supremum over $\mathcal{B}(0, D)$ with an arbitrary M_ℓ-Lipschitz function, which can be done by direct reduction:

Proof of Theorem 29. To apply this to ℓ'', defined on $\mathcal{B}(0, D)$ and M_ℓ Lipschitz, we apply Corollary 10 to $\varphi(x) = \frac{1}{M_\ell D}\ell''(Dx)$ (which is 1-Lipschitz on $\mathcal{B}(0, 1)$). Then, $B_\varphi = B_\ell/M_\ell D$ and the right hand side must be multiplied by $M_\ell D$. □

Remark 4. Note that there is a difference in the way Theorem 32 and Theorem 33 are applied to our linear models problem. In particular, Theorem 32 considers $\varphi_1 = \|\cdot\|^2$ and $\varphi_2 = \ell''$, whereas Theorem 33 uses $\varphi_1 = \varphi_2 = Id$ and $\varphi_3 = \ell''$. Theorem 32 can be adapted to work with $r = 3$, but the bound does not improve when splitting $\|\cdot\|^2$ into $Id \times Id$. Similarly, Theorem 33 could be used with $r = 2$ and $\varphi_1 = \|\cdot\|^2/(2R)$ (to respect the 1-Lipschitz assumption), but in this case the bound can only be worse since the main difference is that the ρ^3 factor becomes $R\rho^2$, and ρ is generally smaller than R.

5.B.4. Tightness of Theorem 33. Consider that the a_i uniformly distributed on the sphere with radius $R = \sqrt{d}$, and take for f_k the identity. Such vectors can be constructed by taking vectors A_i with coordinates $i.i.d.$ standard gaussian, and setting $a_i = \sqrt{d}\|A_i\|^{-1}A_i$. The subgaussianity parameter ρ can then be taken equal to 1.

Using known results about maximal correlation between variables with fixed marginals [e.g., Vershynin, 2019, Section 3], the expectation $\mathbb{E}\prod_{k=1}^r a_1^\top x_k$ is maximized, over choices $x_k \in \mathcal{S}_1$, by taking $x_1 = \cdots = x_r$. We may choose $x_1 = e_1$, the first unit vector, by rotational invariance, and thus the expectation is upper-bounded as:

$$\mathbb{E}\prod_{k=1}^r a_1^\top x_k \leq \mathbb{E}d^{r/2}\mathbb{E}\left[\frac{|A_i(1)|^r}{\|A_i\|^r}\right].$$

This is of order 1, as can be shown using concentration inequalities on the deviations of $\|A_i\|$ from $\sqrt{d}$. Consider then the empirical sum $\frac{1}{n}\sum_{i\in[n]}\prod_{k\in[r]} a_i^\top x_k$. Choose $x_k = d^{-1/2}a_1$ for all $k \in [r]$. Then this empirical sum evaluates to

$$\frac{1}{n}\sum_{i\in[n]}\prod_{k\in[r]} a_i^\top x_k = \frac{1}{n}d^{r/2} + \frac{1}{n}\sum_{i=2}^n\prod_{k\in[r]} a_i^\top a_1.$$

The second sum can be shown to be of order 1 (conditioning on a_1, and then using, *e.g.*, Bienaymé-Tchebitchev inequality). Thus, one cannot hope to establish concentration without extra assumptions on the data distribution unless $d^{r/2} = O(n)$.

Contrast this with the result of Theorem 33: for $R = \sqrt{d}$, $B = R^r$ and $\rho = 1$, it gives that

$$Y \leq O(d^{(r/2)(1-2/r)} d/n) = O(d^{r/2}/n).$$

Thus the result is sharp for the particular example we just considered.

5.C. Experiment Setting and Additional Results

Some implementation details are omitted in the main text. To ease the reader's understanding, we provide these details here, along with some additional experimental results.

Optimization problem. We used the logistic loss with quadratic regularization, meaning that the function at node i is:

$$f_i : x \mapsto \frac{1}{m} \sum_{j=1}^{m} \log\left(1 + \exp(-y_{i,j} x^\top a_j^{(i)})\right) + \frac{\lambda}{2}\|x\|^2,$$

where $y_{i,j} \in \{-1, 1\}$ is the label associated with $a_j^{(i)}$, the j-th sample of node i. The local datasets are constructed by shuffling the LibSVM datasets, and then assigning a fixed portion to each worker. Then, the server subsamples n points from its local dataset to construct the preconditioning dataset. To assess the suboptimality, we let the best algorithm run for more time in order to get a good approximation of the minimum error. Then, we subtract it to the running error of an algorithm to get the suboptimality at each step.

Tuning μ. We tune the base value of μ by starting from $0.1/n$ and then decreasing it as long as it is stable, or increasing it as long as it is unstable. We multiply or divide μ by a factor of 1.2 at each time.

Adjusting α_t and β_t. We found that choosing $A_0 = 0$ and $B_0 = 1$ for SPAG is usually not the best choice. Indeed, rates are asymptotic and sequences α_t and β_t converge very slowly when σ_{rel} is small, whereas we typically rarely use more than about 100 iterations of SPAG. Therefore, we start the algorithm with A_{t_0} and B_{t_0} with $t_0 > 0$ instead. We used $t_0 = 50$, but SPAG is not very sensitive to this choice.

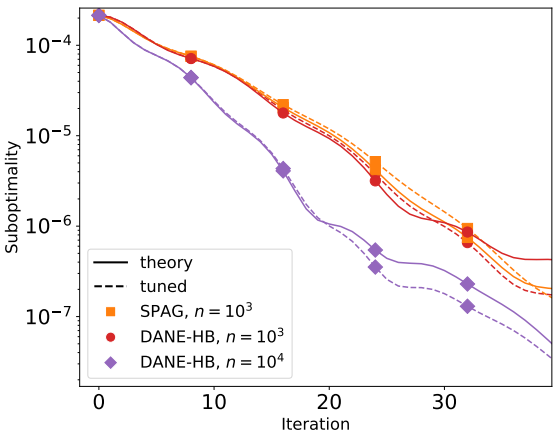
Tuning the momentum. Figure 2(a) evaluates the relevance of tuning the parameters controlling the momentum of SPAG and HB-DANE. To do so, we compare the default values of $\beta = (1 - (1 + 2\mu/\lambda)^{-1/2})^2$ (for HB-DANE) and $\sigma_{\text{rel}} = 1/(1 + 2\mu/\lambda)$ (for SPAG) to values obtained through a grid search on the KDD2010 dataset with $\lambda = 10^{-7}$. We tune HB-DANE by using a grid-search of resolution 0.05 to test the values between 0.5 and 1. For $n = 10^3$, theory predicts a momentum of $\beta = 0.86$ and the grid search gives $\beta = 0.85$. For $n = 10^4$, theory predicts $\beta = 0.81$ and the grid search gives $\beta = 0.8$. For SPAG, we test $\sigma_{\text{rel}} = 10^{-2}$, 3×10^{-3}, 10^{-3} and so on until $\sigma_{\text{rel}} = 10^{-5}$ (so roughly divided by 3 at each step). For $n = 10^3$, theory predicts $\sigma_{\text{rel}} = 0.005$ and the tuning yields $\sigma_{\text{rel}} = 0.006$. For $n = 10^4$, theory predicts $\sigma_{\text{rel}} = 0.0099$ and the grid-search leads to $\sigma_{\text{rel}} = 0.01$. We do not display the curves in this case ($n = 10^4$) since they are nearly identical. Therefore, the grid-search always obtains the value on the grid that is closest to the theoretical value of the parameter, and the difference in practice is rather small, as can be seen in Figure 2(a). This is why we use default values in the main text.

Local subproblems. Local problems are solved using a sparse implementation of SDCA [Shalev-Shwartz, 2016]. In practice, the ill-conditionned regime is very hard, especially when μ is small. Indeed, the local subproblems are very hard to solve, and it should be beneficial to use accelerated algorithms to solve the inner problems. In our experiments, we warm-start the local problems (initializing on the solution of the previous one), and keep doing passes over the preconditioning dataset until $\|\nabla V_t(x_t)\| \leq 10^{-9}$ (checked at each epoch). This threshold is important because it greatly affects the performances of preconditioned gradient methods. Figure 2(b) compares the performances of SPAG, DANE and HB-DANE for different number of passes on the inner problems for the RCV1 dataset for $n = 10^4$ and $\lambda = 10^{-5}$. We use $\mu = 2 \times 10^{-5}$ and a step-size of 1 for all algorithms. We first see that increasing the number of passes significantly improves the convergence speed of all algorithms. Besides, heavy-ball acceleration does not seem very efficient when local problems are not solved accurately enough. On the contrary, SPAG seems to enjoy faster rates than DANE nevertheless. It would be interesting to understand these different behaviours more in details.

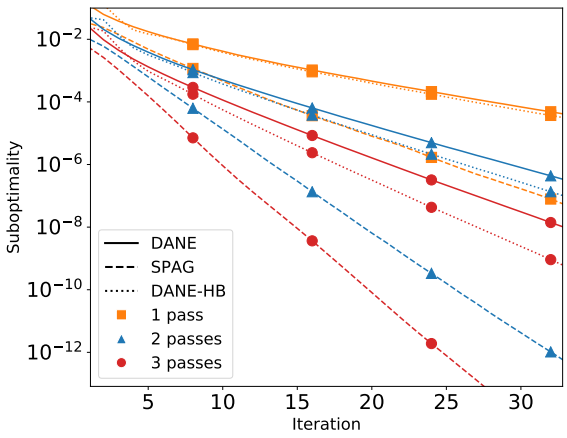
Gain far from the optimum. So far, we have presented experiments with good initializations (solution for the local dataset), and argued why G_t was very small in this case. Because of Lemma 30, one would expect that G_t could be large when x_t is very far from x_*. Yet, We see in the proof of Lemma 30 that the Lipschitz constant of the Hessian only needs to be considered for any convex set that contains x_{t+1}, v_{t+1}, y_t and v_t. In the case of logistic regression, the third derivative decreases very fast when far from 0, meaning that the local Lipschitz constant of the Hessian is small when the iterates are far from 0. In other words, the Hessian changes slowly when far from the optimum (at least for logistic regression).

We believe that this is the reason why G_t can always be chosen of order 1 (smaller than 2) in our experiments, and that this holds regardless of the initialization. To support this claim, we plot in Figure 2(c) the values of the gain for the RCV1 dataset with $\lambda = 10^{-7}$ and 5 different x_0 sampled from $\mathcal{N}(0, 10^3)$, the normal law centered at 0 with variance 10^3. We use a step-size of 0.9 and $\mu = 2 \times 10^{-5}$. We first see that for $G_{\min} = 1$, the gain is always very low, and actually increases at some point instead of becoming lower and lower, so the fact that we were able to choose G_t of order 1 in the other experiments is not linked to the good initialization. We had to choose a slightly higher μ than in the other experiments in order to satisfy the relative smoothness condition, which was not satisfied at each iteration otherwise. Since G_t is small in practice and the smaller the G_t the better the rate, we test SPAG with no minimum value for the gain G_t. The curve for the gain in this case is shown by $G_{\min} = 0$, and we see that the true gain stabilizes to a higher value, since updates are more agressive. We discuss the efficiency of this version in the next paragraph. Note that the oscillations are not due to numerical instability or inaccurate solving of the inner problems, but rather to the fact that the step-size is slightly too big so sometimes the smoothness inequality is not verified. Yet, this does not affect the convergence of SPAG, as shown in Figure 2(d).

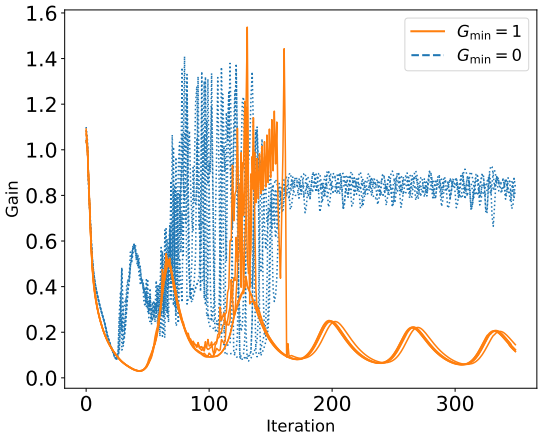
Line Search with no minimum value. Since the gain is almost always smaller than 1, the line-search in SPAG generally only consists in checking that $G_t = 1$ works, which can be done locally. Therefore, there is no added communication cost. As discussed earlier, it is possible to allow $G_t < 1$ when performing line search, which makes SPAG slightly more adaptative at the cost of a few more line-search loops. Figure 2(d) presents the difference



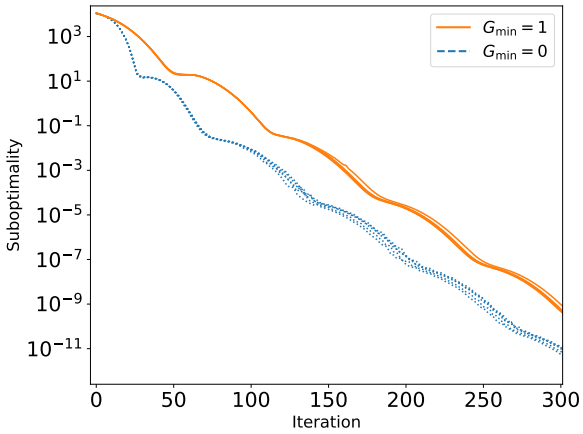
(a) Impact of parameters tuning (KDD2010, $\lambda = 10^{-7}$).



(b) Impact of innacurate solving of the inner problems.



(c) Impact of $G_{\min}$ on the gain.



(d) Impact of $G_{\min}$ on the suboptimality.

FIGURE 2. Impact of several implementation details.

between SPAG using a line search with $G_{\min} = 0$ and $G_{\min} = 1$. The curves show the suboptimality for the runs used to generate Figure 2(c). Note that we omit the cost of line search in the iteration cost (we still count in terms of number of iterations, even though more communication rounds are actually needed when $G_{\min} = 0$). We see that setting $G_{\min} = 0$ is initially slightly faster but that the rate is very similar, so that using $G_{\min} = 0$ may slightly improve iteration complexity but is not worth doing in this case. Note that suboptimality curves for different initializations are almost indistinguishable, which can be explained by the fact that the quadratic penalty term dominates and that all initializations have roughly the same norm (since d is high).

CHAPTER 6

Quantifying the natural differential privacy guarantees of gossip protocols.

Gossip protocols (also called rumor spreading or epidemic protocols) are widely used to disseminate information in massive peer-to-peer networks. These protocols are often claimed to guarantee privacy because of the uncertainty they introduce on the node that started the dissemination. But is that claim really true? Can the source of a gossip safely hide in the crowd? In this Chapter, we examine gossip protocols through a rigorous mathematical framework based on differential privacy to determine the extent to which the source of a gossip can be traceable. Considering the case of a complete graph in which a subset of the nodes are curious, we study a family of gossip protocols parameterized by a "muting" parameter s: nodes stop emitting after each communication with a fixed probability $1 - s$. We first prove that the standard push protocol, corresponding to the case $s = 1$, does not satisfy differential privacy for large graphs. In contrast, the protocol with $s = 0$ (nodes forward only once) achieves optimal privacy guarantees but at the cost of a drastic increase in the spreading time compared to standard push, revealing an interesting tension between privacy and spreading time. Yet, surprisingly, we show that some choices of the muting parameter s lead to protocols that achieve an optimal order of magnitude in both privacy and speed. Privacy guarantees are obtained by showing that only a small fraction of the possible observations by curious nodes have different probabilities when two different nodes start the gossip, since the source node rapidly stops emitting when s is small. The speed is established by analyzing the mean dynamics of the protocol, and leveraging concentration inequalities to bound the deviations from this mean behavior. We also confirm empirically that, with appropriate choices of s, we indeed obtain protocols that are very robust against concrete source location attacks (such as maximum a posteriori estimates) while spreading the information almost as fast as the standard (and non-private) push protocol. This Chapter is based on *Who started this rumor? Quantifying the natural differential privacy of gossip protocols* [Bellet, Guerraoui, and Hendrikx, 2020], published at DISC 2020.

6.1. Introduction

Gossip protocols (also called *rumor spreading* or *epidemic protocols*), in which participants *randomly* choose a neighbor to communicate with, are both simple and efficient means to exchange information in P2P networks Frieze and Grimmett [1985], Pittel [1987], Karp et al. [2000], Berenbrink et al. [2010]. They are a basic building block to propagate and aggregate information in distributed databases Demers et al. [1987], Boyd et al. [2006] and social networks Doerr et al. [2011], Giakkoupis et al. [2015], to model the spread of infectious diseases Hethcote [2000], as well as to train machine learning models on distributed datasets Duchi et al. [2012a], Colin et al. [2016], Vanhaesebrouck et al. [2017], Koloskova et al. [2019b].

Some of the information gossiped may be sensitive, and participants sharing it may not want to be identified. This can for instance be the case of whistle-blowers or individuals that would like to exercise their right to freedom of expression in totalitarian regimes. Conversely, it may sometimes be important to locate the source of a (computer or biological) virus, or fake news, in order to prevent it from spreading before too many participants get "infected".

There is a folklore belief that gossip protocols inherently guarantee some form of *source anonymity* because participants cannot know who issued the information in the first place Ghaffari and Newport [2016]. Similarly, identifying "patient zero" for real-world epidemics is known to be a very hard task. Intuitively indeed, random and local exchanges make identification harder. But to what extent? Although some work has been devoted to the design of source location strategies in specific settings Jiang et al. [2017], Pinto et al. [2012], Shah and Zaman [2011], the general anonymity claim has never been studied from a pure *privacy* perspective, that is, independently of the very choice of a source location technique. Depending on the use-case, it may be desirable to have strong privacy guarantees (e.g., in anonymous information dissemination) or, on the contrary, we may hope for weak guarantees, e.g., when trying to identify the source of an epidemic. In both cases, it is crucial to precisely quantify the anonymity level of gossip protocols and study its theoretical limits through a principled approach. This is the challenge we take up in this paper for the classic case of gossip dissemination in a complete network graph.

Our first contribution is an information-theoretic model of anonymity in gossip protocols based on (ϵ, δ)-*differential privacy* (DP) Dwork [2006], Dwork et al. [2006]. Originally introduced in the database community, DP is a precise mathematical framework recognized as the gold standard for studying the privacy guarantees of information release protocols. In our proposed model, the information to protect is the source of the gossip, and an adversary tries to locate the source by monitoring the communications (and their relative order) received by a subset of f curious nodes. In a computer network, these curious nodes may have been compromised by a surveillance agency; in our biological example, they could correspond to health professionals who are able to identify whether a given person is infected. Our notion of DP then requires that the probability of any possible observation of the curious nodes is almost the same no matter who is the source, thereby limiting the predictive power of the adversary regardless of its actual source location strategy. A distinctive aspect of our model is that the mechanism that seeks to ensure DP comes only from the *natural* randomness and partial observability of gossip protocols, not from additional perturbation or noise which affects the desired output as generally needed to guarantee DP Dwork and Roth [2014]. We believe our adaptation of DP to the gossip context to be of independent interest. We also complement it with a notion of *prediction uncertainty* which guarantees that even unlikely events do not fully reveal the identity of the source under a uniform prior on the source. This property directly upper bounds the probability of success of any source location attack, including the maximum likelihood estimate.

We use our proposed model to study the privacy guarantees of a generic family of gossip protocols parameterized by a *muting parameter* s: nodes have a fixed probability $1-s$ to stop emitting after each communication (until they receive the rumor again). In our biological parallel, this corresponds to the fact that a person stops infecting other people after some time. The muting parameter captures the ability of the protocol to forget initial conditions, thereby helping to conceal the identity of the source. In the extreme case where $s = 1$, we

recover the standard "push" gossip protocol Pittel [1987], and show that it is inherently *not* differentially private for large graphs. In contrast, we also show that, at the other end of the spectrum, choosing $s = 0$ leads to *optimal privacy guarantees* among all gossip protocols.

More generally, we determine *matching upper and lower bounds* on the privacy guarantees of gossip protocols. Essentially, our upper bounds on privacy are obtained by tightly lower bounding the probability that the source node contacts a curious node before another node does, and upper bounding the probability that this happens for a random node fixed in advance, in a way that holds for all gossip algorithms. Remarkably, despite the fact that the source node always has a non-negligible probability of telling the rumor to a curious node first, our results highlight the fact that setting $s = 0$ leads to strong privacy guarantees in several regimes, including the pure $(\epsilon, 0)$-DP as well as prediction uncertainty.

It turns out that, although achieving optimal privacy guarantees, choosing $s = 0$ leads to a very slow spreading time (*log-linear* in the number of nodes n). This highlights an interesting tension between *privacy* and *spreading time*: the two extreme values for the muting parameter s recover the two extreme points of this trade-off. We then show that more balanced trade-offs can be achieved: appropriate choices of the muting parameter lead to gossip protocols that are *near-optimally private* with a spreading time that is *logarithmic* in the size of the graph. In particular, the trade-off between privacy and speed shows up in the constants but, surprisingly, some choices of the parameter lead to protocols that achieve an optimal order of magnitude for both aspects. Our results on this trade-off are summarized in Table 1: for a proportion f/n of curious nodes, one can see that setting the muting parameter $s = f/n$ achieves almost optimal privacy (up to a factor 2) while being substantially faster than $s = 0$ (optimal up to a factor f/n). Similarly, if one wants to achieve $(0, \delta_0)$-differential privacy with $\delta_0 > 2f/n$, then it is possible to set $s = \delta_0/2$ and obtain a protocol that respects the privacy constraint with spreading time $\mathcal{O}(\log(n)/\delta_0)$. From a technical perspective, these privacy results are obtained by showing that only a small fraction of the possible observations by curious nodes have different probabilities when two different nodes start with the gossip. This requires to precisely evaluate the probability of well-chosen worst-case sequences, which is generally hard as randomness is involved both when nodes decide to stop spreading the rumor (with probability $1-s$) and when they choose who to communicate with. Our parameterized gossip protocol can be seen as a population protocol Angluin et al. [2008], and we prove its speed by analyzing its mean dynamics and leveraging concentration inequalities to bound the deviations from the mean dynamics.

We support our theoretical findings by an empirical study of our parameterized gossip protocols. The results show that appropriate choices of s lead to protocols that are very robust against classical source location attacks (such as maximum a posteriori estimates) while spreading the information almost as fast as the standard (and non-private) push protocol. Crucially, we observe that our differential privacy guarantees are very well aligned with the ability to withstand attacks that leverage background information, e.g., targeting known activists or people who have been to certain places.

The rest of the paper is organized as follows. We first discuss related work and formally introduce our concept of differential privacy for gossip. Then, we study two extreme cases of our parameterized gossip protocol: the standard push protocol, which we show is not private, and a privacy-optimal but slow protocol. This leads us to investigate how to better

	Muting param.	δ ensuring $(0, \delta)$-DP	Spreading time
Standard push (minimal privacy, maximal speed)	$s = 1$	1	$\mathcal{O}(\log n)$
Muting after infecting (maximal privacy, minimal speed)	$s = 0$	$\frac{f}{n}$	$\mathcal{O}(n \log n)$
Generic parameterized gossip (privacy vs. speed trade-off)	$0 < s < 1$	$s + (1 - s)\frac{f}{n}$	$\mathcal{O}(\log(n)/s)$

TABLE 1. Summary of results to illustrate the tension between privacy and speed. n is the total number of nodes and f/n is the fraction of curious nodes in the graph. $\delta \in [0, 1]$ quantifies differential privacy guarantees (smaller is better). Spreading time is asymptotic in n.

control the trade-off between speed and privacy. Finally, we present our empirical study and conclude by discussing open questions.

For pedagogical reasons, we keep our model relatively simple to avoid unnecessary technicalities in the derivation and presentation of our results. For completeness, we discuss the impact of possible extensions (e.g., information observed by the adversary, malicious behavior, termination criterion) in Appendix 6.A. For space limitations, some detailed proofs are also deferred to the appendix.

6.2. Background and Related Work

6.2.1. Gossiping. The idea of disseminating information in a distributed system by having each node *push* messages to a randomly chosen neighbor, initially coined the *random phone-call model*, dates back to even before the democratization of the Internet Pittel [1987]. Such protocols, later called *gossip*, *epidemic* or *rumor spreading*, were for instance applied to ensure the consistency of a replicated database system Demers et al. [1987]. They have gained even more importance when argued to model spreading of infectious diseases Hethcote [2000] and information dissemination in social networks Doerr et al. [2011], Giakkoupis et al. [2015]. Gossip protocols can also be used to compute aggregate queries on a database distributed across the nodes of a network Kempe et al. [2003a], Boyd et al. [2006], and have recently become popular in federated machine learning Kairouz et al. [2019] to optimize cost functions over data distributed across a large set of peers Duchi et al. [2012a], Colin et al. [2016], Vanhaesebrouck et al. [2017], Koloskova et al. [2019b]. Gossip protocols differ according to their interaction schemes, i.e., *pull* or *push*, sometimes combining both Karp et al. [2000], Kowalski and Caro [2013], Acan et al. [2017].

In this work, we focus on the classical *push* form in the standard case of a *complete* graph with n nodes (labeled from 0 to $n - 1$). We now define its key communication primitive. Denoting by I the set of informed nodes, `tell_gossip`(i, I) allows an informed node $i \in I$ to tell the information to another node $j \in \{0, ..., n - 1\}$ chosen uniformly at random. `tell_gossip`(i, I) returns j (the node that received the message) and the updated I (the new set of informed nodes that includes j). Equipped with this primitive, we can now formally define the class of gossip protocols that we consider in this paper.

DEFINITION 7 (Gossip protocols). *A gossip protocol on a complete graph is one that (a) terminates almost surely, (b) ensures that at the end of the execution the set of informed nodes I is equal to $\{0, ..., n-1\}$, and (c) can modify I only through calls to* `tell_gossip`.

6.2.2. Locating the Source of the Gossip. Determining the source of a gossip is an active research topic, especially given the potential applications to epidemics and social networks, see Jiang et al. [2017] for a recent survey. Existing approaches have focused so far on building *source location attacks* that compute or approximate the maximum likelihood estimate of the source given some observed information. Each approach typically assumes a specific kind of graphs (e.g., trees, small world, *etc.*), dissemination model and observed information. In *rumor centrality* Shah and Zaman [2011], the gossip communication graph is assumed to be fully observed and the goal is to determine the *center* of this graph to deduce the node that started the gossip. Another line of work studies the setting in which some nodes are *curious sensors* that inform a central entity when they receive a message Pinto et al. [2012]. Gossiping is assumed to happen at random times and the source node is estimated by comparing the different timings at which information reaches the sensors. The proposed attack is natural in trees but does not generalize to highly connected graphs. The work of Fanti et al. [2017] focuses on hiding the source instead of locating it. The observed information is a snapshot of who has the rumor at a given time. A specific dissemination protocol is proposed to hide the source but the privacy guarantees only hold for tree graphs.

We emphasize that the privacy guarantees (i.e., the probability not to be detected) that can be derived from the above work only hold under the specific attacks considered therein. Furthermore, all approaches rely on maximum likelihood and hence assume a uniform prior on the probability of each node to be the source. The guarantees thus break if the adversary knows that some of the nodes could not have started the rumor, or if he is aware that the protocol is run twice from the same source.

We note that other problems at the intersection of gossip protocols and privacy have been investigated in previous work, such as preventing unintended recipients from learning the rumor Georgiou et al. [2011], and hiding the initial position of agents in a distributed system Gotfryd et al. [2017].

6.2.3. Differential Privacy. While we borrow ideas from the approaches mentioned above (e.g., we assume that a subset of nodes are curious sensors as in Pinto et al. [2012]), we aim at studying the fundamental limits of *any* source location attack by measuring the amount of information leaked by a gossip scheme about the identity of the source. For this purpose, a general and robust notion of privacy is required. *Differential privacy* Dwork [2006], Dwork and Roth [2014] has emerged as a gold standard for it holds independently of any assumption on the model, the computational power, or the background knowledge that the adversary may have. Differentially private protocols have been proposed for numerous problems in the fields of databases, data mining and machine learning: examples include computing aggregate and linear counting queries Dwork and Roth [2014], releasing and estimating graph properties Lu and Miklau [2014], Sun et al. [2019a], clustering Huang and Liu [2018], empirical risk minimization Chaudhuri et al. [2011] and deep learning Abadi et al. [2016].

In this work, we consider the classic version of differential privacy which involves two parameters $\epsilon, \delta \geq 0$ that quantify the privacy guarantee Dwork et al. [2006]. More precisely,

given any two databases $\mathcal{D}_1$ and $\mathcal{D}_2$ that differ in at most one record, a randomized information release protocol $\mathcal{P}$ is said to guarantee (ϵ, δ)-differential privacy if for any possible output S:

$$p(\mathcal{P}(\mathcal{D}_1) \in S) \leq e^{\epsilon} p(\mathcal{P}(\mathcal{D}_2) \in S) + \delta, \tag{6.2.1}$$

where $p(E)$ denotes the probability of event E. Parameter ϵ places a bound on the ratio of the probability of any output when changing one record of the database, while parameter δ is assumed to be small and allows the bound to be violated with small probability. When $\epsilon = 0$, δ gives a bound on the total variation distance between the output distributions, while $\delta = 0$ is sometimes called "pure" ϵ-differential privacy. DP guarantees hold regardless of the adversary and its background knowledge about the records in the database. In our context, the background information could be the knowledge that the source is among a subset of all nodes. Robustness against such background knowledge is crucial in some applications, for instance when sharing secret information that few people could possibly know or when the source of an epidemic is known to be among people who visited a certain place. Another key feature of differential privacy is *composability*: if (ϵ, δ)-differential privacy holds for a release protocol, then querying this protocol two times about the same dataset satisfies $(2\epsilon, 2\delta)$-differential privacy. This is important for rumor spreading as it enables to quantify privacy when the source propagates multiple rumors that the adversary can link to the same source (e.g., due to the content of the message). We will see in Section 6.6 that these properties are essential in practice to achieve robustness to concrete source location attacks.

Existing differentially private protocols typically introduce additional *perturbation* (also called *noise*) to hide critical information Dwork and Roth [2014]. In contrast, an original aspect of our work is that we will solely rely on the *natural* randomness and limited observability brought by gossip protocols to guarantee differential privacy.

6.3. A Model of Differential Privacy for Gossip Protocols

Our first contribution is a precise mathematical framework for studying the fundamental privacy guarantees of gossip protocols. We formally define the inputs of the gossip protocols introduced in Definition 7, the outputs observed by the adversary during their execution, and the privacy notions we investigate. To ease the exposition, we adopt the terminology of information dissemination, but we sometimes illustrate the ideas in the context of epidemics.

6.3.1. Inputs and Outputs. As described in Section 6.2.3, differential privacy is a probabilistic notion that measures the privacy guarantees of a protocol based on the variations of its *output* distribution for a change in its *input*. In this paper, we adapt it to our gossip context. We first formalize the *inputs* and *outputs* of gossip protocols, in the case of a *single piece of information to disseminate* (multiple pieces can be addressed through composition, see Section 6.2.3). At the beginning of the protocol, a single node, the source, has the information (the gossip, or rumor). This node defines the input of the gossip protocol, and it is the actual "database" that we want to protect. Therefore, in our context, input databases in Equation (6.2.1) have only 1 record, which contains the identity of the source (an integer between 0 and $n-1$). Therefore, all possible input databases differ in at most one record, and differential privacy aims at protecting the content of the database, i.e., which node started the rumor.

We now turn to the outputs of a gossip protocol. The execution of a protocol generates an ordered sequence S_{omni} of pairs (i, j) of calls to `tell_gossip` where $(S_{\text{omni}})_t$ corresponds to the t-th time the `tell_gossip` primitive has been called, i is the node on which `tell_gossip` was used and j the node that was told the information. If several calls to `tell_gossip` happen simultaneously, ties are broken arbitrarily. We assume that the messages are received in the same order that they are sent. This protocol can thus be seen as an epidemic population protocol model Angluin et al. [2008] in which nodes interact using `tell_gossip`. The sequence S_{omni} corresponds to the output that would be observed by an omniscient entity who could eavesdrop on all communications. It is easy to see that, for any execution, the source can be identified exactly from S_{omni} simply by retrieving $(S_{\text{omni}})_0$.

In this work, we focus on adversaries that monitor a set of *curious nodes* $\mathcal{C}$ of size f, i.e. they observe all communications involving a curious node. This model, previously introduced in Pinto et al. [2012], is particularly meaningful in large distributed networks: while it is unlikely that an adversary can observe the full state of the network at any given time, compromising or impersonating a subset of the nodes appears more realistic. The number of curious nodes is directly linked with the *release mechanism* of DP: while the final state of the system is always the same (everyone knows the rumor), the information released depends on which messages were received by the curious nodes during the execution. Formally, the output disclosed to the adversary during the execution of the protocol, i.e., the information he can use to try to identify the source, is a subsequence of S_{omni} as defined below.

Assumption 12. *The sequence S observed by the adversary through the (random) execution of the protocol is a (random) subsequence $S = ((S_{\text{omni}})_t | (S_{\text{omni}})_t = (i, j)$ with $j \in \mathcal{C})$, that contains all messages sent to curious nodes. The adversary has access to the relative order of tuples in S, which is the same as in S_{omni}, but not to the index t in S_{omni}.*

It is important to note that the adversary does not know which messages were exchanged between non-curious nodes. In particular, he does not know how many messages were sent in total at a given time. As we focus on complete graphs, knowing which curious node received the rumor gives no information on the source node. For a given output sequence S, we write $S_t = i$ to denote that the t-th `tell_gossip` call in S originates from node i. Omitting the dependence on S, we also denote $t_i(j)$ the time at which node j first receives the message (even for the source) and $t_d(j)$ the time at which j first communicates with a curious node.

The ratio f/n of curious nodes determines the probability of the adversary to gather information (the more curious nodes, the more information leaks). For a fixed f, the adversary only becomes weaker as the network grows bigger. Since we would like to study adversaries with fixed power, unless otherwise noted we make the following assumption.

Assumption 13. *The ratio of curious nodes f/n is constant.*

Finally, we emphasize that we do not restrict the computational power of the adversary.

6.3.2. Privacy Definitions. We now formally introduce our privacy definitions. The first one is a direct application of differential privacy (Equation 6.2.1) for the inputs and outputs specified in the previous section. To ease notations, we denote by I_0 the source of the gossip (the set of informed nodes at time 0), and for any given $i \in \{0, ..., n-1\}$, we denote by $p_i(E) = p(E|I_0 = \{i\})$ the probability of event E if node i is the source of the

gossip. The protocol is therefore abstracted in this notation. Denoting by $\mathcal{S}$ the set of all possible outputs, we say that a gossip protocol is (ϵ, δ)-differentially private if:

$$p_i(S) \leq e^{\epsilon} p_j(S) + \delta, \quad \forall S \subset \mathcal{S}, \ \forall i, j \in \{0, ..., n-1\}, \tag{6.3.1}$$

where $p(S)$ is the probability that the output belongs to the set S. This formalizes a notion of *source indistinguishability* in the sense that any set of output which is likely enough to happen if node i starts the gossip (say, $p_i(S) \geq \delta$) is almost as likely (up to a e^{ϵ} multiplicative factor) to be observed by the adversary regardless of the source. Note however that when $\delta > 0$, this definition can be satisfied for protocols that release the identity of the source (this can happen with probability δ). To capture the behavior under unlikely events, we also consider the complementary notion of *c-prediction uncertainty* for $c > 0$, which is satisfied if for a uniform prior $p(I_0)$ on source nodes and any $i \in \{0, ..., n-1\}$:

$$p(I_0 \neq \{i\}|S)/p(I_0 = \{i\}|S) \geq c, \tag{6.3.2}$$

for any $S \subset \mathcal{S}$ such that $p_i(S) > 0$. Prediction uncertainty guarantees that no observable output S (however unlikely) can identify a node as the source with large enough probability: it ensures that the probability of success of any source location attack is upper bounded by $1/(1+c)$. This holds in particular for the maximum likelihood estimate. Prediction uncertainty does not have the robustness of differential privacy against background knowledge, as it assumes a uniform prior on the source. While it can be shown that $(\epsilon, 0)$-DP with $\epsilon > 0$ implies prediction uncertainty, the converse is not true. Indeed, prediction uncertainty is satisfied as soon as no output identifies any node with enough probability, without necessarily making all pairs of nodes indistinguishable as required by DP. We will see that prediction uncertainty allows to rule out some naive protocols that have nonzero probability of generating sequences which reveal the source with certainty.

Thanks to the symmetry of our problem, we consider without loss of generality that node 0 starts the rumor ($I_0 = \{0\}$) and therefore we will only need to verify Equations (6.3.1) and (6.3.2) for $i = 0$ and $j = 1$.

6.4. Extreme Privacy Cases

In this section, we study the fundamental limits of gossip in terms of privacy. To do so, we parameterize gossip protocols by a muting parameter $s \in [0, 1]$, as depicted in Algorithm 11. We thereby capture, within a generic framework, a large family of protocols that fit Definition 7 and work as follows. They maintain a set A of *active nodes* (initialized to the source node) which spread the rumor asynchronously and in parallel: this is modeled by the fact that at each step of the protocol, a randomly selected node $i \in A$ invokes the `tell_gossip` primitive to send the rumor to another node (which in turn becomes active), while i also stays active with probability s. This is illustrated in Figure 1. The muting parameter s can be viewed as a randomized version of *fanout* in Eugster et al. [2004].[1] Algorithm 11 follows the population protocol model Angluin et al. [2008], and is also related to the SIS epidemic model Hethcote [2000] but in which the rumor never dies regardless of the value of $s \in [0, 1]$ (there always remain some active nodes). Although we present it from a centralized

[1] Unlike in the classic fanout, nodes start to gossip again each time they receive a message instead of deactivating permanently.

Algorithm 11 Parameterized Gossip

Require: n // *Number of nodes*, k // *Source node*, s // *Probability for a node to remain active*

Ensure: $I = \{0, \dots, n-1\}$ // *All nodes are informed*

$I \leftarrow \{k\}$, $A \leftarrow \{k\}$
while $|I| < n$ **do**
  Sample i uniformly at random from A
  $A \leftarrow A \setminus \{i\}$ with probability $1 - s$
  $j, I \leftarrow$ `tell_gossip`(i, I), $A \leftarrow A \cup \{j\}$

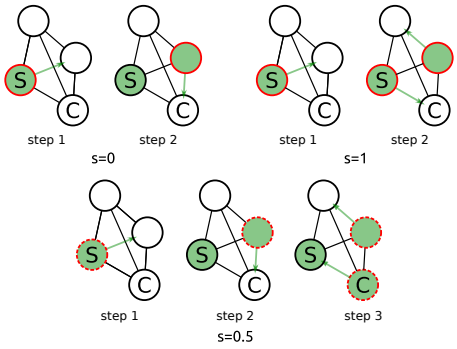


FIGURE 1. *Left:* Parameterized Gossip. *Right:* Illustration of the role of muting parameter s. `S` indicates the source and `C` a curious node. Green nodes know the rumor, and red circled nodes are active. When $s = 0$, there is only one active node at a time, which always stops emitting after telling the gossip. In the case $s = 1$, nodes always remain active once they know the rumor (this is the standard push gossip protocol Pittel [1987]). When $0 < s < 1$, each node remains active with probability s after each communication.

perspective, we emphasize that Algorithm 11 is asynchronous and can be implemented by having active nodes wake up following a Poisson process.

In the rest of this section, we show that extreme privacy guarantees are obtained for extreme values of the muting parameter s.

6.4.1. Standard Push has Minimal Privacy. The natural case to study first in our framework is when the muting parameter is set to $s = 1$: this corresponds to the standard push protocol Pittel [1987] in which nodes always keep emitting after they receive the rumor. Theorem 34 shows that, surprisingly, the privacy guarantees of this protocol become arbitrarily bad as the size of the graph increases (keeping the fraction of curious nodes constant).

THEOREM 34 (Standard push is not differentially private). *If Algorithm 11 with $s = 1$ guarantees (ϵ, δ)-DP for all values of n and constant $\epsilon < \infty$, then $\delta = 1$.*

This result may seem counter-intuitive at first since one could expect that it would be more and more difficult to locate the source when the size of the graph increases. Yet, since the ratio of curious nodes is kept constant, this result comes from the fact that the event $\{t_d(0) \leq t_i(1)\}$ (node 0 communicates with a curious node before node 1 gets the message) becomes more and more likely as n grows, hence preventing any meaningful differential privacy guarantee when n is large enough. The proof is in Appendix 6.C.1. Theorem 34 clearly highlights the fact that the standard gossip protocol ($s = 1$) is not differentially private in general. We now turn to the other extreme case, where the muting parameter $s = 0$.

6.4.2. Muting After Infecting has Maximal Privacy. We now study the privacy guarantees of generic Algorithm 11 when $s = 0$. In this protocol, nodes forward the rumor to exactly one random neighbor when they receive it and then stop emitting until they receive the rumor again. Intuitively, this is good for privacy: the source changes and it is quickly impossible to recover which node started the gossip (as initial conditions are quickly forgotten). In fact, once the source tells the rumor once, the state of the system (the set of active nodes, which in this case is only one node) is completely independent from the source. A similar idea was used in the protocol introduced in Fanti et al. [2017].

The following result precisely quantifies the privacy guarantees of Algorithm 11 with parameter $s = 0$ and shows that it is *optimally private* among all gossip protocols (in the precise sense of Definition 7).

THEOREM 35. *Let $\epsilon \geq 0$. For muting parameter $s = 0$, Algorithm 11 satisfies (ϵ, δ)-differential privacy with $\delta = \frac{f}{n}\left(1 - \frac{e^\epsilon - 1}{f}\right)$ and c-prediction uncertainty with $c = \frac{n}{f+1} - 1$. Furthermore, these privacy guarantees are optimal among all gossip protocols.*

PROOF OF THEOREM 35. We start by proving the fundamental limits on the privacy of any gossip protocol, and then prove matching guarantees for Algorithm 11 with $s = 0$.

(Fundamental limits in privacy) Proving a lower bound on the differential privacy parameters can be achieved by finding a set of possible outputs S (here, a set of ordered sequences) such that $p_0(S) \geq p_1(S)$. Indeed, a direct application of the definition of Equation (6.3.1) yields that given any gossip protocol, $S \subset \mathcal{S}$ and $w_0, w_1 \in \mathbb{R}$ such that $w_0 \leq p_0(S)$ and $p_1(S) \leq w_1$, if the protocol satisfies (ϵ, δ) differential privacy then $\delta \geq w_0 - e^\epsilon w_1$. The proofs need to consider all the messages sent and then distinguish between the ones that are disclosed (sent to curious nodes) and the ones that are not.

Since $I = \{0\}$ then `tell_gossip` is called for the first time by node 0 and it is called at least once otherwise the protocol terminates with $I = \{0\}$, violating the conditions of Definition 7. We denote by $S^{(0)}$ the set of output sequences such that $S_0 = 0$ (i.e., 0 is the first to communicate with a curious node). We also define the event $T_0^c = \{t_d(0) \neq 0\}$ (the source does not send its first message to a curious node). For all $i \notin \mathcal{C} \cup \{0\}$, we have that $p_0(S_0 = i|T_0^c) \leq p_0(S_0 = 0|T_0^c)$ since $p_0(A_1 = \{0\}) = p_0(i \in A_1)$, where A_1 is the set of active nodes at time 1. From this inequality we get

$$\textstyle\sum_{i\notin\mathcal{C}} p_0(S_0 = 0|T_0^c) \geq \sum_{i\notin\mathcal{C}} p_0(S_0 = i|T_0^c) = 1 \geq \sum_{i\notin\mathcal{C}} p_0(S_0 = 1|T_0^c),$$

where the equality comes from the fact that $S_0 = i$ for some $i \notin \mathcal{C}$. The second inequality comes from the fact that $p_j(S_0 = i|T_0^c) = p_j(S_0 = k|T_0^c)$ for all $i, k \neq j$. Therefore, we have $p_0(S_0 = 0|T_0^c) \geq \frac{1}{n-f}$ and $p_0(S_0 = 1|T_0^c) \leq \frac{1}{n-f}$. Combining the above expressions, we derive the probability of $S^{(0)}$ when 0 started the gossip. We write $p_0(S^{(0)}) = p_0(S^{(0)}, t_d(0) = 0) + p_0(S^{(0)}, T_0^c)$ and then, since $p_0\Big(S^{(0)}|t_d(0) = 0\Big) = 1$:

$$p_0\Big(S^{(0)}\Big) = p_0\Big(t_d(0) = 0\Big)p_0\Big(S^{(0)}|t_d(0) = 0\Big) + p_0\Big(S^{(0)}|T_0^c\Big)p_0\Big(T_0^c\Big) \geq \frac{f}{n} + \frac{1}{n-f}\Big(1 - \frac{f}{n}\Big)$$

In the end, $p_0(S^{(0)}) \geq \frac{f}{n} + \frac{1}{n}$. If node 1 initially has the message, we do the same split and obtain $p_1(S^{(0)}|t_d(0) = 0) = 0$ and so $p_1(S^{(0)}) = p_1(T_0^c)p_1(S^{(0)}|T_0^c) \leq \frac{1}{n}$.

The upper bound on prediction uncertainty is derived using the same quantities:

$$\frac{p(I_0 \neq 0|S^{(0)})}{p(I_0 = 0|S^{(0)})} = \sum_{i \notin C \cup \{0\}} \frac{p_i(S^{(0)})}{p_0(S^{(0)})} \leq (n-f-1)\frac{p_1(S^{(0)})}{p_0(S^{(0)})} \leq \frac{n-f-1}{f+1} = \frac{n}{f+1} - 1.$$

Note that we have never assumed that curious nodes knew how many messages were sent at a given point in time. We have only bounded the probability that the source is the first node that sends a message to curious nodes.

(Matching guarantees for Algorithm 11 with $s = 0$) For this protocol, the only outputs S such that $p_0(S) \neq p_1(S)$ are those in which $t_d(0) = 0$ or $t_d(1) = 0$. We write:

$$p_0(S_0 = 0) = p_0(t_d(0) = 0)p_0(S_0 = 0|t_d(0) = 0) + p_0(T_0^c)p_0(S_0 = 0|T_0^c).$$

For any $i \notin \mathcal{C}$ where $\mathcal{C}$ is the set of curious nodes, we have that $p_0(S_0 = 0|T_0^c) = p_0(S_0 = i|T_0^c) = \frac{1}{n-f}$. Indeed, given that $t_d(0) \neq 0$, the node that receives the first message is selected uniformly at random among non-curious nodes, and has the same probability to disclose the gossip at future rounds. Plugging into the previous equation, we obtain:

$$p_0(S_0 = 0) = \frac{f}{n} + \left(1 - \frac{f}{n}\right)\frac{1}{n-f} = \frac{f+1}{n}.$$

For any other node $i \notin \mathcal{C} \cup \{0\}$, $p_0(S_0 = i) = p_0(T_0^c)p_0(S_0 = i|T_0^c) = \frac{1}{n}$ because $p_0(S_0 = i|t_d(0) = 0) = 0$. Combining these results we get $p_0(S^{(0)}) \leq e^\epsilon p_1(S^{(0)}) + \delta$ for any $\epsilon > 0$ and $\delta = \frac{f}{n}(1 - \frac{e^\epsilon - 1}{f})$. By symmetry, we make a similar derivation for $S^{(1)}$.

To prove the prediction uncertainty result, we use the differential privacy result with $e^\epsilon = f + 1$ (and thus $\delta = 0$) and write that for any $S \in \mathcal{S}$:

$$\frac{p(I_0 \neq 0|S)}{p(I_0 = 0|S)} = \sum_{i \notin C \cup \{0\}} \frac{p_i(S)}{p_0(S)} \geq (n-f-1)e^{-\epsilon} = \frac{n}{f+1} - 1. \qquad \square$$

Theorem 35 establishes *matching upper and lower bounds* on the privacy guarantees of gossip protocols. More specifically, it shows that setting the muting parameter to $s = 0$ provides strong privacy guarantees that are in fact optimal. Note that in the regime where $\epsilon = 0$ (where DP corresponds to the total variation distance), δ cannot be smaller than the proportion of curious nodes. This is rather intuitive since the source node has probability at least f/n to send its first message to a curious node. However, one can also achieve differential privacy with δ much smaller than f/n by trading-off with $\epsilon > 0$. In particular, the *pure* version of differential privacy ($\delta = 0$) is attained for $\epsilon \approx \log f$, which provides good privacy guarantees when the number of curious nodes is not too large. Furthermore, even though the probability of disclosing *some* information is of order f/n, prediction uncertainty guarantee shows that an adversary with uniform prior always has a high probability of making a mistake when predicting the source. Crucially, these privacy guarantees are made possible by the *natural* randomness and partial observability of gossip protocols.

Remark 5 (Special behavior of the source). *A subtle but key property of Algorithm 11 is that the source follows the same behavior as other nodes. To illustrate how violating this property may give away the source, consider this natural protocol: the source node transmits the rumor to one random node and stops emitting, then standard push (Algorithm 11 with $s = 1$) starts from the node that received the information. While this* delayed start gossip protocol *achieves optimal differential privacy in some regimes, it is fundamentally flawed.*

In particular, it does not guarantee prediction uncertainty in the sense that $c \to 0$ as the graph grows. Indeed, the adversary can identify the source with high probability by detecting that it communicated only once and then stopped emitting for many rounds. We refer to Appendix 6.B for the formal proof.

6.5. Privacy vs. Speed Trade-offs

While choosing $s = 0$ achieves optimal privacy guarantees, an obvious drawback is that it leads to a very slow protocol since only one node can transmit the rumor at any given time. It is easy to see that the number of gossip operations needed to inform all nodes can be reduced to the time needed for the classical coupon collection problem: it takes $\mathcal{O}(n \log n)$ communications to inform all nodes with probability at least $1 - 1/n$ Erdős and Rényi [1961]. As this protocol performs exactly one communication at any given time, it needs time $\mathcal{O}(n \log n)$ to inform all nodes with high probability. This is in stark contrast to the standard push gossip protocol ($s = 1$) studied in Section 6.4.1 where all informed nodes can transmit the rumor in parallel, requiring only time $\mathcal{O}(\log n)$ Frieze and Grimmett [1985].

These observations motivate the exploration of the privacy-speed trade-off (with parameter $0 < s < 1$). We first show below that nearly optimal privacy can be achieved for small values of s. Then, we study the spreading time and show that the $\mathcal{O}(\log n)$ time of the standard gossip protocol also holds for $s > 0$, leading to a sweet spot in the privacy-speed trade-off.

6.5.1. Privacy Guarantees. Theorem 36 conveys a $(0, \delta)$-differential privacy result, which means that apart from some unlikely outputs that may disclose the identity of the source node, most of these outputs actually have the same probability regardless of which node triggered the dissemination. We emphasize that the guarantee we obtain holds for any graph size with fixed proportion f/n of curious nodes.

THEOREM 36 (Privacy guarantees for $s < 1$). *For $0 < s < 1$ and any fixed $r \in \mathbb{N}^*$, Algorithm 11 with muting parameter s guarantees $(0, \delta)$-differential privacy with:*

$$\delta = 1 - (1 - s) \sum_{k=0}^{\infty} s^k \left(1 - \frac{f}{n}\right)^{k+1} \leq 1 - (1 - s^r) \left(1 - \frac{f}{n}\right)^r .$$

For example, choosing $r = 1$ leads to $\delta \leq s + (1 - s)\frac{f}{n}$, as reported in Table 1. Slightly tighter bounds can be obtained, but this is enough already to recover optimal guarantees as $s \to 0$.

PROOF. We first consider that S is such that $t_d(0) \geq t_d(1)$. Then, $p_0(S) \leq p_1(S)$ since node 0 needs to receive the rumor before being able to communicate it to curious nodes, and Equation (6.3.1) is verified. Suppose now that S is such that $t_d(0) \leq t_d(1)$. In this case, we note t_m the first time at which the source stops to emit (which happens with probability $1-s$ each time it sends a message). Then, we denote $F = \{t_d(0) \leq t_m\}$ (and F^c its complement). In this case, $p_0(S|F^c) \leq p_1(S|F^c)$. Indeed, conditioned on F^c, $t_d(0) \geq t_i(0)$ if node 0 is not the source and $t_d(0) \geq \max(t_m, t_i(0))$ if it is. Then, we can write:

$$p_0(S) = p_0(S, F^c) + p_0(S, F) \leq p_1(S, F^c) + p_0(F) \leq p_1(S) + p_0(F).$$

Denoting T_f the number of messages after which the source stops emitting, we write:

$$p_0(F) = \sum_{k=1}^{\infty} p_0(T_f = k) p_0(F | T_f = k) = \sum_{k=0}^{\infty} (1-s)s^k \Big(1 - \Big(1 - \frac{f}{n}\Big)^{k+1}\Big), \text{ for } s > 0.$$

Note that we can also write for $k \geq 1$ that $p_0(F) = p_0(F, T_f \leq k) + p_0(F, T_f > k)$, and so:

$$p_0(F) \leq (1-s^k)\Big(1 - \Big(1 - \frac{f}{n}\Big)^k\Big) + s^k = 1 - (1-s^k)\Big(1 - \frac{f}{n}\Big)^k. \qquad \square$$

The differential privacy guarantees given by Theorem 36 and the optimal guarantees of Theorem 35 are of the same order of magnitude when s is of order f/n. Indeed, consider $\epsilon = 0$. Then, setting $r = 1$ in Theorem 36 leads to an additive gap of $s(1 - f/n)$ between the privacy of Algorithm 11 and the optimal guarantee, showing that one can be as close as desired to the optimal privacy as long as s is chosen close enough to 0. In particular, the ratio between the privacy of Algorithm 11 and the lower bound is less than 2 for all $s \leq f/n$. This indicates that the privacy guarantees are very tight in this regime. We also recover exactly the optimal guarantee of Theorem 35 in the case $s = 0$ (without the ability to control the trade-off between ϵ and δ). Importantly, we also show that Algorithm 11 with $s < 1$ satisfies prediction uncertainty, unlike the case where $s = 1$.

THEOREM 37. *Algorithm 11 guarantees prediction uncertainty with* $c = (1 - \frac{f+1}{n})(1-s)$.

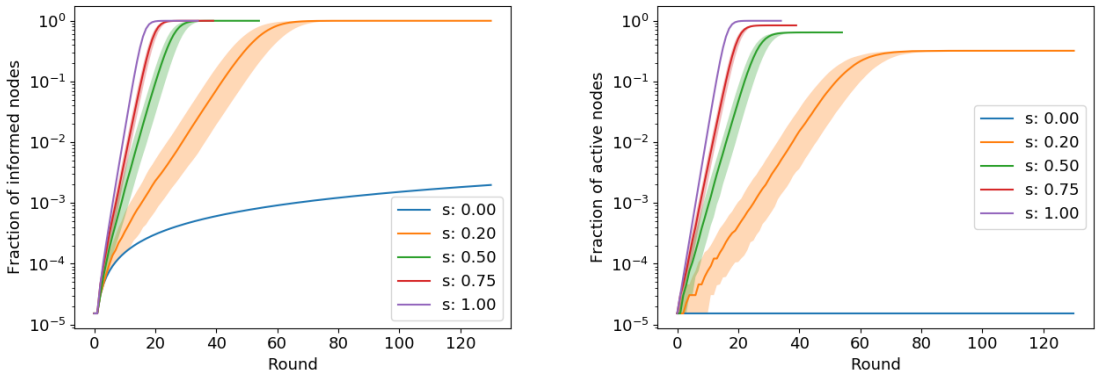
This result is another evidence that picking $s < 1$ allows to derive meaningful privacy guarantees. The proof can be found in Appendix 6.C.1.

6.5.2. Spreading time. We have shown that parameter s has a significant impact on privacy, from optimal ($s = 0$) to very weak ($s = 1$) guarantees. Yet, s also impacts the spreading time: the larger s, the more active nodes at each round. This is highlighted by the two extreme cases, for which the spreading time is already known and exhibits a large gap: $\mathcal{O}(\log n)$ for $s = 1$ and $\mathcal{O}(n \log n)$ for $s = 0$. To establish whether we can obtain a protocol that is both private and fast, we need to characterize the spreading time for the cases where $0 < s < 1$.

The key result of this section is to prove that the logarithmic speed of the standard push gossip protocol holds more generally for all $s > 0$. This result is derived from the fact that the ability to forget does not prevent an *exponential growth* phase. What changes is that the population of active nodes takes approximately $1/s$ rounds to double instead of 1 for standard gossip. For ease of presentation, we state below the result for the synchronous version of Algorithm 11, in which the notion of *round* corresponds to iterating over the full set A. A similar result (with an appropriate notion of rounds) can be obtained for the asynchronous version given in Algorithm 11.

THEOREM 38. *For a given* $s > 0$ *and for all* $1 > \delta > 0$ *and* $C \geq 1$, *there exists* n *large enough such that the synchronous version of Algorithm 11 with parameter* s *sends at least* $Cn \log n$ *messages in* $6C \log(n)/s$ *rounds with probability at least* $1 - \delta$.

PROOF SKETCH. The key argument of the proof is that the gossip process very closely follows its mean dynamics. After a transition phase of a logarithmic number of rounds, a constant fraction of the nodes (depending on s) remains active despite the probability to stop emitting after each communication. This "determinism of gossip process" has been



(a) Fraction of informed nodes

(b) Fraction of active nodes

FIGURE 2. Effect of parameter s of Algorithm 11 on the spreading time for a network of $n = 2^{16}$ nodes. The curves represent median values and the shaded area represents the 10 and 90 percent confidence intervals over 100 runs. Each curve stops when all nodes are informed (and so the protocol terminates), except for $s = 0$ since the protocol is very slow in this case.

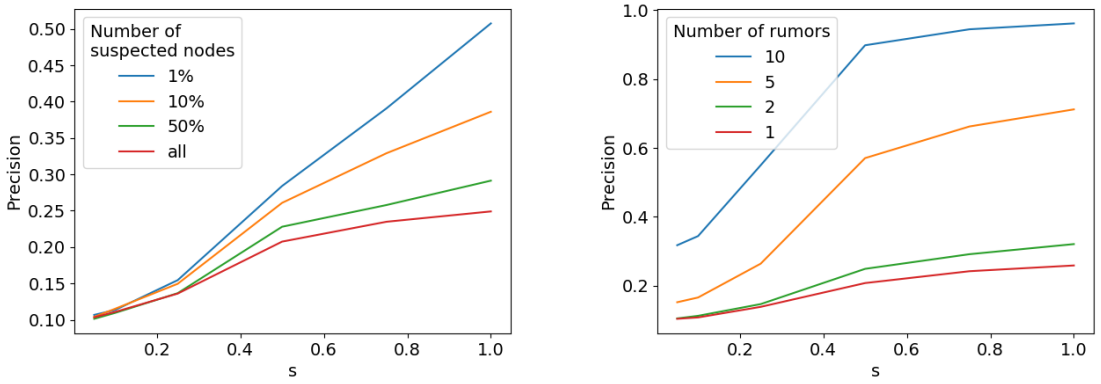
introduced in Sanghavi et al. [2007], but their analysis only deals with the case $s = 1$. Our proof takes into account the nontrivial impact of nodes deactivation in the exponential and linear growth phase. Besides, we need to introduce and analyze a last phase, showing that with high probability the population never drops below a critical threshold of active nodes. The full proof is in Appendix 6.C.2. □

Theorem 38 shows that generic gossip with $s > 0$ still achieves a logarithmic spreading time even though nodes can stop transmitting the message. The $1/s$ dependence is intuitive since $1/s$ rounds are needed in expectation to double the population of active nodes (without taking collisions into account). Therefore, the exponential growth phase which usually takes time $\mathcal{O}(\log n)$ now takes time $\mathcal{O}(\log(n)/s)$ for $s < 1$. To summarize, we have shown that one can achieve both fast spreading and near-optimal privacy, leading to the values presented in Table 1 of the introduction.

6.6. Empirical Evaluation

In this section, we evaluate the practical impact of s on the spreading time as well as on the robustness to source location attacks run by adversaries with background knowledge.

6.6.1. Spreading Time. To complement Theorem 38, which proves logarithmic spreading time (asymptotic in n), we run simulations on a network of size $n = 2^{16}$. The logarithmic spreading time for $s > 0$ is clearly visible in Figure 2(a), where we see that the gossip spreads almost as fast for $s = 0.5$ that it does for $s = 1$. We also observe that even when s is small, the gossip remains much faster than for $s = 0$. The results in Figure 2(b) illustrate that the fraction of active nodes grows exponentially fast for all values of $s > 0$ and then reaches a plateau when the probability of creating a new active node is compensated by the probability



(a) Attack precision under prior information on the source

(b) Attack precision when the source spreads multiple rumors

FIGURE 3. Effect of parameter s of Algorithm 11 on the precision of source location attacks for a network of $n = 2^{16}$ node with 10% of curious nodes. Precision is estimated over 15,000 random runs.

of informing an already active node. Empirically, this happens when the fraction of active nodes is of order s.

We note incidentally that gossip protocols are often praised for their robustness to lost messages Alistarh et al. [2010], Georgiou et al. [2013]. While the protocol with $s = 0$ does not tolerate a single lost message, setting $s > 0$ improve the resilience thanks to the linear proportion of active nodes. The latter property makes it unlikely that the protocol stops because of lost messages as long as s is larger than the probability of losing messages. Of course, the protocol remains somewhat sensitive to messages lost during the first few steps.

6.6.2. Robustness Against Source Location Attacks. Getting an intuitive understanding of the privacy guarantees provided by Theorem 36 is not straightforward, as often the case with differential privacy. Therefore, we illustrate the effect of the muting parameter on the guarantees of our gossip protocol by simulating concrete source location attacks. We consider two challenging scenarios where the adversary has some background knowledge: either 1) prior knowledge that the source belongs to a subset of the nodes, or 2) side information indicating that the same source disseminates multiple rumors.

Prior knowledge on the source. We first consider the case where the adversary is able to narrow down the set of suspected nodes. In this case we can design a provably optimal attack, as shown by the following theorem (see Appendix 6.C.3 for the proof).

THEOREM 39. *If the adversary has a uniform prior over a subset P of nodes, i.e., $p(I_0 = i) = p(I_0 = j)$ for all $i, j \in P$ and $p(I_0 = i) = 0$ for $i \notin P$, and for some output sequence S, t_c is such that $S_{t_c} \in P$ and $S_t \notin P$ if $t < t_c$, then $p(I_0 = S_{t_c}|S) \geq p(I_0 = i|S)$ for all i.*

Theorem 39 means that under a uniform prior over nodes in P, the attack in which curious nodes predict the source to be the first node in P that communicates with them corresponds to the Maximum A Posteriori (MAP) estimator. The set P represents the prior knowledge of the adversary: he knows for sure that the source belongs to P.

Figure 3(a) shows the precision of this attack as a function of s for varying degrees of prior knowledge. We see that, when s is small, the prior knowledge does not improve the attack precision significantly, and that the precision remains very close to the probability that the source sends its first message to a curious node. This robustness to prior knowledge is consistent with the properties of differential privacy (see Section 6.2.3). On the contrary, when s is high (i.e., differential privacy guarantees are weak), the impact of the prior knowledge on the precision of the attack is much stronger.

Multiple dissemination. We investigate another scenario in which differential privacy guarantees can also provide robustness, namely when the adversary knows that the same source node disseminates multiple rumors. In this setting, analytically deriving an optimal attack is very difficult. Instead, we design an attack which leverages the fact that even though the source is not always the first node to communicate with curious nodes, with high probability it will be among the first to do so. More precisely, the curious nodes record the 10 first nodes that communicate with them in each instance (results are not very sensitive to this choice), and they predict the source to be the node that appears in the largest number of instances. In case of a tie, the curious nodes choose the node that first communicated with them, with ties broken at random. Figure 3(b) shows that the precision of this attack increases dramatically with the number of rumors when s is large, reaching almost sure detection for 10 rumors. Remarkably, for small values of s, the attack precision increases much more gracefully with the number of rumors, as expected from the composition property of differential privacy discussed in Section 6.2.3. Meaningful privacy guarantees can still be achieved as long as the source does not spread too many rumors.

6.7. Concluding Remarks

This paper initiates the formal study of privacy in gossip protocols to determine to which extent the source of a gossip can be traceable. Essentially: (1) We propose a formal model of anonymity in gossip protocols based on an adaptation of differential privacy; (2) We establish tight bounds on the privacy of gossip protocols, highlighting their natural privacy guarantees; (3) We precisely capture the trade-off between privacy and speed, showing in particular that it is possible to design both fast and near-optimally private gossip protocols; (4) We experimentally evaluate the speed of our protocols as well as their robustness to source location attacks, validating the relevance of our formal differential privacy guarantees in scenarios where the adversary has some background knowledge.

Our work opens several interesting perspectives. In particular, it paves the way to the study of differential privacy for gossip protocols in *general graphs*, which is challenging and requires relaxations of differential privacy in order to obtain nontrivial guarantees. We refer to Appendix 6.D for a discussion of these questions. Another avenue for future research is motivated by very recent work showing that hiding the source of a message can amplify differential privacy guarantees for the *content* of the message Erlingsson et al. [2018], Cheu et al. [2018], Balle et al. [2019]. Unfortunately, classic primitives to hide the source of messages such as mixnets can be difficult and costly to deploy. Showing that gossip protocols can *naturally* amplify differential privacy for the message contents would make them very desirable for privacy-preserving distributed AI applications, such as those based on federated Kairouz et al. [2019] and decentralized machine learning Bellet et al. [2018].

6.A. Model Extensions

We kept our model relatively simple in the main paper to avoid unnecessary complexity in the notations and additional technicalities in the derivation and presentation of our results. In this appendix, we briefly discuss various possible extensions. Basically, we make here the point that, although these generally lead to technical complications, they do not significantly impact privacy guarantees.

6.A.1. Pull and Push-Pull Protocols. Our study focus on the classic *push* form of gossip protocols. This can be justified by the fact that, for regular graphs, synchronous push has asymptotic spreading time guarantees that are comparable with the push-pull variant Giakkoupis et al. [2016]. Besides, the differential privacy guarantees of any gossip protocol are limited by the probability that the first node informed by the source is a curious node, and we show this bound can be matched with push protocols. Nevertheless, extensions of our results to pull and push-pull variants of gossip protocols Karp et al. [2000] are possible. Forgetting mechanisms similar to the ones in Algorithm 11 can be introduced for these protocols, i.e. nodes would have a probability $1 - s$ to stop disclosing information after each time they are pulled (if they do not pull someone with the information in between). Although slightly different, the optimal privacy guarantees would remain of the same order of magnitude. Yet, we expect pull guarantees to be even worse in the case $s = 1$ because curious nodes could stop suspecting all nodes that they have pulled and that did not have the rumor. Besides, the pull protocol for $s = 0$ would be even slower than its push counterpart.

6.A.2. Eavesdropping Adversary. Since we consider a complete graph, our formalization of the adversary as a fraction f/n of curious nodes is closely related to an eavesdropping adversary who would observe each communication with probability f/n. Indeed, both models consider that each communication has a probability f/n of being disclosed to the adversary. Most of our results are thus easily transferable to this alternative setting. The only difference would be that all nodes can be suspected in the eavesdropping model, thus introducing a $(1 - f/n)^{-1}$ factor each time we consider the population of non-curious nodes.

6.A.3. Information Observed by the Adversary. We discuss three possible variants of the output observed by the adversary.

Messages Sent by Curious Nodes. For simplicity of exposition, we considered that curious nodes only observe messages that are sent to them and not the messages that they send. However, including the messages sent by curious nodes in their observed output would not impact the bounds on privacy (i.e., the guarantees for the algorithms). For the optimal algorithm, we only consider what happens during the first round, so including the messages sent by curious nodes does not change the result. This in particular implies that the fundamental limits of Theorem 35 remain the same (since the adversary observes strictly more information). Similarly, for the parameterized algorithm, Theorem 36 is obtained by bounding the probability of a set $\hat{S}$. Then, we have $p(\hat{S}, S_{\text{out}}) \leq p(\hat{S})$ where S_{out} is the sequence of messages sent by the curious nodes. In general, adding the messages sent by curious nodes to the output sequences has little or no impact on our results.

Message Ordering. We assumed that the relative order of messages is preserved in the output sequence observed by curious nodes. This could be relaxed, as in real-world networks a message sent before another may well be received after it. One could for instance introduce

a random swapping model to take this into account and investigate whether this weaker output leads to an improvement in the privacy guarantees. However, we argue that this improvement would be quite limited. First of all, it would not affect the privacy guarantees of the optimal protocol: since there is a single active node able to send a message at any given time, swapping is not possible. Therefore, the lower bound and the matching algorithm would not be affected by this change. Since parameterized gossip is almost privacy-optimal for small values of s and swapping would only increase privacy, then we argue that the guarantees of parameterized gossip would be very similar in this case. Furthermore, even when several nodes are active at the same time (e.g., in Algorithm 11 large s), the proofs can be adapted to work with counting the messages *received* instead of the messages *sent*. In this case, swapping is as likely to expose the source (making its messages arrive earlier) than to hide it (delaying the messages it sends).

Global Timing. In our model, we assume that curious nodes only have access to the relative ordering in which they received the messages but they have no information on the global time at which it was sent. This is justified in practice by the asynchrony and locality of the exchanges. We briefly discuss here how the privacy guarantees are affected if one considers a stronger adversary that has access to the number of times the `tell_gossip` procedure has been called. Formally, this adversary observes the set $S = \{(t, i, j) | (i, j) = (S_{\text{omni}})_t, j \in \mathcal{C}\}$. This set can be turned into a sequence by ordering it by increasing values of t. Note that this is not a realistic adversary as gossip protocols naturally enforce partial observability of the events.

The following result quantifies the limits of privacy for this stronger adversary, which can be compared to the results of Theorem 35 in the main text. We can see that in the regime $\epsilon = 0$ (total variation distance), the limits remain the same. However, achieving $\delta < f/n$ and prediction uncertainty is not possible against this stronger adversary. Note also that Algorithm 1 with $s = 0$ remains optimal.

THEOREM 40. *If a gossip protocol satisfies (ϵ, δ)-differential privacy and c-prediction uncertainty then we have $\delta \geq \frac{f}{n}$ and $c = 0$ in the strong adversary setting. Furthermore, these bounds are tight and matched by Algorithm 1 when its parameter is set to $s = 0$.*

PROOF. The fact that `tell_gossip` is called at least once and is first called on node 0 still holds. Sequence $S^{(0)}$ now denotes the fact that node 0 communicates with a curious node at time 0. Since the protocol is run on the complete graph, the node selected by `tell_gossip` is chosen uniformly within $\{0, ..., n-1\}$, so a curious node is selected with probability $\frac{f}{n}$. We thus have $p_0(S^{(0)}) = \frac{f}{n}$. Besides, node 0 cannot communicate with a curious node at time 0 if node 1 starts the rumor so $p_1(S^{(0)}) = 0$. For prediction uncertainty, using the same sequence $S^{(0)}$ yields $\frac{p_i(S^{(0)})}{p_0(S^{(0)})} = 0$ for all $i \neq 0$ and therefore $c = 0$.

It remains to show that these bounds are matched by Algorithm 11 with $s = 0$. The fact that the only outputs that have a different probability if node 0 starts (compared to the case when 1 starts) are those in which 0 (or 1) communicates with a curious node for its first communication is still true with the stronger adversary. Then, we write $p_0(S_0 = 0) = p_1(S_0 = 1) = \frac{f}{n}$ and $p_0(S_0 = 1) = p_1(S_0 = 0) = 0$. This ensures that $p_0(S^{(0)}) \leq p_1(S^{(0)}) + \frac{f}{n}$ (similarly for $S^{(1)}$), and the result follows. □

6.A.4. Malicious Behavior. We also assumed for simplicity that nodes are *curious* but not *malicious*, i.e., they follow the protocol. This is motivated by a practical scenario where a subset of nodes are simply being monitored by a curious entity. If curious nodes can also act maliciously, they have three possible ways to affect the protocol (abstracting away the content of the information): emitting more, emitting less, or not choosing neighbors uniformly at random. If they emit more, they will inform more nodes, which makes it more difficult for them to locate the source. If they emit less (potentially not at all), then in the case $s < 1$, the protocol could stop before all nodes are informed. Yet, the privacy bounds are derived from the fact that the source forgets the information before communicating to a curious node. Choosing the neighbors they send the messages to reduces to the case in which they emit less (for they do not send messages to uninformed nodes) but without affecting protocol speed or termination (this does not reduce the number of active nodes). Thus, the impact on the observed output and therefore on the privacy would be minimal. In the case $s = 1$, malicious nodes have slightly more impact but remain quite small: this case only makes the set of informed nodes grow slightly slower.

6.A.5. Termination Criterion. For simplicity, in all our gossip protocols we used a global termination criterion (the protocol terminates when all nodes are informed). Termination without using global coordination is a problem in its own right that has been extensively studied (see for instance Karp et al. [2000]). Although some termination criteria could have a great impact on privacy, we argue that termination can be handled late in the execution so as to reveal very little about the beginning, hence avoiding any significant impact on privacy. For instance, it is possible to design a variant of Algorithm 11 in which nodes only flip a coin with probability s for a fixed number of times, and then stop emitting completely. This fixed number would have to depend on s, but then if it is large enough, it would guarantee both termination and privacy. Indeed, nodes would not communicate with curious nodes each time they are activated with high probability so this counter would actually provide very little information to the curious nodes. Determining how large this number of iterations should be, and the exact impact on privacy (which we argue is very small), is beyond the scope of this paper.

6.B. Delayed Start Gossip

Consider the protocol described in Remark 5, which we call *delayed start gossip*:

1. The source calls `tell_gossip` once to forward to an arbitrary node, say node j.
2. Node j then starts a standard push protocol (Algorithm 11 with $s = 1$).

This simple protocol is clearly optimal from the point of view of differential privacy in the regime $\epsilon = 0$ (total variation distance). Indeed, if the first communication does not hit a curious node then the probability of a given output when two different nodes start the gossip is the same. It is also fast since it runs the standard gossip after the first round.

Yet, this naive protocol has a major flaw. Indeed, when the first communication hits a curious node, the adversary can monitor whether the sender communicates with curious nodes again in the next rounds. If it does not, they can guess that the node is the source, and they will in fact make a correct guess with probability arbitrarily close to 1 for large enough graphs. On the other hand, when the sender communicates again with a curious node shortly after, they can be very confident that this node is not the source. Hence, it

is possible to design a very simple attack with a very high precision (almost always right) and almost optimal recall (identifying the source with certainty every time the information is actually released, i.e. with probability $\frac{f}{n}$).

Making sure that the adversary is uncertain about its prediction is therefore a desirable property. This is captured by our notion of *prediction uncertainty*. The following proposition formalizes the above claims.

PROPOSITION 1. *We call c_{ds} the prediction uncertainty constant of the delayed start protocol and we assume the ratio of curious nodes f/n to be constant. Then $c_{ds} \to 0$ when $n \to \infty$.*

More generally, it is in principle possible to prove similar results for any protocol in which the source node does not behave like other nodes. Indeed, if the special behaviour can be detected, then the adversary can know for sure the source of the rumor. This motivates the need for more involved protocols such as those covered by Algorithm 11.

PROOF OF PROPOSITION 1. The proof reuses some elements of the proof of Theorem 34. We consider the sequence $S_r^{(0)}$ such that node 0 is the first node to communicate with a curious node ($S_0 = 0$) and then r other nodes communicate with curious nodes before 0 does ($S_i \neq 0$ for $i \in \{1, ..., r\}$). We denote by t_0 the time at which node 0 gets the message and becomes active again (we refer here to the global order, although of course the curious nodes do not have access to it). Then, with the usual notations we have:

$$\begin{aligned} p_0\left(S_r^{(0)}\right) &= p_0(S_0 = 0) p_0\left(S_r^{(0)} | S_0 = 0\right) \\ &\geq \frac{f}{n} p_0\left(\cap_{i=1}^r S_i \neq 0 | S_0 = 0\right) \\ &\geq \frac{f}{n} p_0(t_0 \geq r) \\ &\geq \frac{f}{n} p_0(n_c(r) \leq k^*) p_0(t_0 \geq r | n_c(r) \leq k^*). \end{aligned}$$

Then, we recall from the proof of Theorem 34 that

$$\begin{aligned} p_0(n_c(r) \leq k) &= p\left(Binom(k, \frac{f}{n}) \geq r\right) \\ &= p\left(Binom(k, 1 - \frac{f}{n}) < k - r\right) \\ &= 1 - p\left(Binom(k, 1 - \frac{f}{n}) \geq k - r\right), \end{aligned}$$

so if we set $k = \frac{2n}{f} r$ and use tail bounds on the binomial law (Theorem 1 of Arratia and Gordon [1989]) then there exists a constant H (only depending on f/n) such that $p_0(n_c(r) \leq r\frac{2n}{f}) \geq 1 - e^{-rH}$. Therefore, we have:

$$p_0\left(S_r^{(0)}\right) \geq \frac{f}{n}\left(1 - e^{-rH}\right)\left(1 - \frac{1}{n}\right)^{r\frac{2f}{n}} \geq C_1(r, n). \tag{6.B.1}$$

The last line comes from calculations done in the proof of Theorem 34. We now study $p_1(S_r^{(0)})$. Since node 1 started the protocol then it means that no other node (and in particular

0) will stop emitting the message. Therefore, if node 0 is the first to communicate with a curious node then it will remain active for the whole duration of the protocol. Consider that the first disclosure happens after T_f communications. We can write:

$$p_1\left(S_r^{(0)}\right) \leq p_1(S_0 = 0)p_1\left(\cap_{i=1}^r S_i \neq 0 | S_0 = 0, T_f \leq t_f\right) + p_1(T_f > t_f).$$

Since the fraction of curious nodes is constant, we can choose t_f independently of n or r such that $p(T_f > t_f) \leq e^{-\frac{f}{n}t_f} \leq \frac{\epsilon}{4(n-f)}$ if $t_f = \frac{n}{f}\log\left(\frac{4(n-f)}{\epsilon}\right)$ in order to control the second term. Then,

$$p_1\left(\cap_{i=1}^r S_i \neq 0 | S_0 = 0, T_f \leq t_f\right) \leq \prod_{t=t_f}^{t_f+r}\left(1 - \frac{f}{n}\frac{1}{t}\right) \leq e^{-\frac{f}{n}\sum_{t=t_f}^{t_f+r}\frac{1}{t}}.$$

A series-integral comparison yields that if $r = \log^2(n)$ then $\exp\left(-\frac{f}{n}\sum_{t=t_f}^{t_f+r}\frac{1}{t}\right) \leq \frac{\epsilon}{4}$ for n large enough. Finally, we use the fact that $p_1(S_0 = 0) \leq \frac{1}{n-f}$ to write that $p_1\left(S_r^{(0)}\right) \leq \frac{\epsilon}{2(n-f)}$.

Finally, we observe that $C_1(\log^2 n, n) \to \frac{f}{n}$ when $n \to \infty$ where C_1 is defined in Equation 6.B.1. In particular, $C_1(\log^2 n, n) \geq \frac{f}{2n}$ for n large enough, so we have

$$\frac{p(I_0 \neq 0 | S_r^{(0)})}{p(I_0 = 0 | S_r^{(0)})} = \sum_{i \notin \mathcal{C}\cup\{0\}} \frac{p_i(S_r^{(0)})}{p_0(S_r^{(0)})} \leq \frac{n}{f}\epsilon. \tag{6.B.2}$$

Since ϵ can be picked arbitrarily small and $\frac{n}{f}$ is assumed to be constant then the previous ratio can be made arbitrary small. □

6.C. Detailed Proofs

6.C.1. Privacy Guarantees.

Proof of Theorem 34. Intuitively, the proof relies on the fact that the event $\{t_d(0) \leq t_i(1)\}$ (node 0 communicates with a curious node before node 1 gets the message) becomes more and more likely as n grows, hence preventing any meaningful differential privacy guarantee when n is large enough. To formalize this, we study $S_r^{(0)} = \{S, S_t = 0 \text{ for some } t \leq r\}$, the set of output sequences such that the rank of node 0 in the sequence is less than r. For a specific sequence to not be in $S_r^{(0)}$, there must have been at least r communications (because r nodes must have communicated with curious nodes), and none of them involved 0 and a curious node. Therefore, if we note $n_c(r)$ the number of communications that actually happened before the output sequence reached size r, we have $n_c(r) \geq r$. Then, denoting by $C(t)$ the node that communicated with a curious node at time t (with $C(t) = -1$ when the communication did not involve a curious node):

$$p_0(S_r^{(0)}) = 1 - p\left(\cap_{t=0}^{n_c(r)} C(t) \neq 0\right) = 1 - \prod_{t=0}^{n_c(r)} p\left(C(t) \neq 0\right) \geq 1 - \prod_{t=0}^{r}\left(1 - \frac{f}{n}\frac{1}{t+1}\right),$$

where the last step comes from the fact that the probability of node 0 to be selected at time t is $\frac{1}{|I_t|} \geq \frac{1}{t}$ because at most one node is informed at each step and the active node is selected uniformly among informed nodes. We use the fact that $\log(1+x) \leq x$ for any $x > -1$ on $x = -\frac{f}{n}\frac{1}{t+1}$ to get:

$$\prod_{t=0}^{r}\left(1 - \frac{f}{n}\frac{1}{t+1}\right) = \exp\left(\sum_{t=0}^{r}\log\left(1 - \frac{f}{n}\frac{1}{t+1}\right)\right) \leq \exp\left(-\frac{f}{n}\sum_{t=0}^{r}\frac{1}{t+1}\right). \tag{6.C.1}$$

Therefore, $p_0(S_r^{(0)})$ goes to 1 as r goes to infinity. We emphasize that we do not need to fix any network size for this result to hold since the ratio f/n is assumed to be constant.

Then, for a given r and for any $k > 0$, $p(n_c(r) \leq k)$ is equal to $p(Binom(k, \frac{f}{n}) \geq r)$ where $Binom(k, \frac{f}{n})$ is the binomial law of parameters k and $\frac{f}{n}$. This is because it is the probability of having exactly r successes with the sum of less than k Bernoullis of parameter $\frac{f}{n}$, which is equal to the probability of having more than r successes with the sum of k Bernoullis of the same parameters. Therefore, $p(n_c(r) \leq k)$ is independent of n and we can choose k^* independently of n such that $p(n_c(r) > k^*) \leq \frac{1}{n}$. Then, we write that

$$p_1(S_r^{(0)}) = p_1(S_r^{(0)}, n_c(r) \leq k^*) + p_1(S_r^{(0)}, n_c(r) > k^*) \leq p_1(S_r^{(0)} | n_c(r) \leq k^*) + 1/n.$$

This implies $p_1(S_r^{(0)} | n_c(r) \leq k^*) \leq p_1(0 \in I_r | n_c(r) \leq k^*) \leq 1 - p_1(0 \notin I_r | n_c(r) \leq k^*)$. We know that only r communications have reached curious nodes but the others have reached a random node in the graph, and there is at most k^* of them, so finally:

$$p_1(S_r^{(0)}) \leq 1 - \left(1 - \frac{1}{n}\right)^{k^*} + \frac{1}{n}.$$

We immediately see that $p_1(S_r^{(0)})$ goes to 0 as n grows because k^* is independent of n, and we have shown above that $p_0(S_r^{(0)})$ goes to 1 as n grows. Since we must have that $p_0(S_r^{(0)}) \leq e^{\epsilon} p_1(S_r^{(0)}) + \delta$, we must have $\delta = 1$ if we want δ and ϵ to be independent of n. □

Proof of Theorem 37. For any set of sequences $S \subset \mathcal{S}$ such that $p_0(S) > 0$:

$$\frac{p(I_0 \neq 0 | S)}{p(I_0 = 0 | S)} = \sum_{i \notin \mathcal{C} \cup \{0\}} \frac{p_i(S)}{p_0(S)} \geq \sum_{i \notin \mathcal{C} \cup \{0\}} \frac{p_i(A_1 = \{0\}) p_i(S | A_1 = \{0\})}{p_0(S)},$$

where A_1 is the set of active nodes at round 1. Because the state of the system (active nodes) is the same in both cases we can write that $p_i(S | A_1 = \{0\}) = p_0(S)$. Besides, $p_i(A_1 = \{0\})$ corresponds to the probability that node i sends a message to node 0 and then stops emitting. Therefore: $\frac{p(I_0 \neq 0 | S)}{p(I_0 = 0 | S)} \geq \left(1 - \frac{f+1}{n}\right)(1 - s) > 0$. □

6.C.2. Spreading time.

Proof of Theorem 38. The idea of this proof is to rely on the "determinism" of gossip process, similarly to Sanghavi et al. [2007]. This means that the gossip process very closely follows its mean dynamics. In our case, there is an added difficulty in the fact that extra randomness is introduced by the deactivation of the nodes. Yet, we precisely quantify the impact of this phenomenon on the results. We start by showing that if more than $k(s)$ nodes are informed at a given time, then with very high probability the number of informed nodes never drops below this fraction once it is reached. Therefore, a number of messages proportional to the size of the graph is sent at each round. The condition on s for this to happen is written in Equation (6.C.6). More formally, we fix $s \in (0, 1]$ and denote by A_t the number of nodes that are active at round t, which is such that $A_t = \alpha_t n$. Then, we note

$$f : \alpha \to 1 - p_u(\alpha)(1 - \alpha s), \tag{6.C.2}$$

where $p_u(\alpha) = (1 - \frac{1}{n})^{\alpha n}$. Note that $f(\alpha) = \frac{1}{n}\mathbb{E}[A_{t+1} | A_t = \alpha n]$. To see this, we count the number of active nodes at time $t + 1$. In total, $A_t = \alpha n$ messages are sent at the beginning of the round. Therefore, for each node, the probability of having received a message at the end of the round is exactly $1 - p_u(\alpha)$ since each message has a $1/n$ probability to be sent to

this specific node. In the end, $n\,(1-p_u(\alpha))$ nodes get the message in expectation. The rest of the active nodes at time $t+1$ is made of the nodes that were active, did not receive the message and did not deactivate, which represents a portion $n\alpha p_u(\alpha)s$ of the nodes. Then, one can see that the function f is simply the sum of these 2 terms. We show by using that $(1-x)^y \leq e^{-xy} \leq 1-xy+\frac{x^2y^2}{2}$ that for $\alpha \leq \alpha_s = \frac{s}{1+2s}$, we have:

$$f(\alpha) \geq \left(1+\frac{s}{2}\right)\alpha. \tag{6.C.3}$$

Then, we follow the same steps as in Lemma 15 in Sanghavi et al. [2007]. We call A_t the number of active nodes at round t, and $A_{t,m}$ the number of active nodes at round t after m messages have been sent (so during the round). Then, we can define $X_i = A_{t,i+1} - A_{t,i}$. $A_{t,i+1}$ only depends on $A_{t,i}$ and so does X_i:

$$X_i = \begin{cases} 1 & \text{with proba} \quad s(1-\frac{|A_{t,i}|}{n}) \\ -1 & \text{with proba} \quad (1-s)\frac{|A_{t,i}|-1}{n} \\ 0 & \text{otherwise} \end{cases}$$

Then, we define the martingale $Z_i = \mathbb{E}[\sum_{i=1}^{A_t} X_i | X_1, \cdots, X_i, A_t]$. This allows us to write $A_{t+1} - nf(\alpha) = Z_0 - Z_{A_t}$. If we call $S_{k,t} = \sum_{i=k}^{A_t} X_i$ then for any $d \in \{-1,0,1\}$:

$$\begin{aligned} &\mathbb{E}[S_{1,t}|X_1,,X_i,X_{i+1}=1,A_t] \\ &\geq \mathbb{E}[S_{1,t}|X_1,\cdots,X_i,X_{i+1}=d,A_t] \\ &\geq \mathbb{E}[S_{1,t}|X_1,\cdots,X_i,X_{i+1}=-1,A_t], \end{aligned}$$

because the distribution of X_i only depends on $A_{t,i}$. Therefore, $|Z_{i+1}-Z_i| \leq (1+\mathbb{E}[S_{i+1,t}|A_t+1]) - (\mathbb{E}[S_{i+1,t}|A_t-1]-1)] \leq 2$. Azuma's inequality Mitzenmacher and Upfal [2005] then gives:

$$p\left(A_{t+1} - nf(\frac{A_t}{n}) \leq -\lambda A_t | A_t = k\right) \leq e^{-\frac{(\lambda k)^2}{8k}}. \tag{6.C.4}$$

We also have that $p(A_{t+1} < k | A_t \geq k) \leq p(A_{t+1} \leq k | A_t = k)$. Then, for any $\alpha \leq \alpha_s$, Equation 6.C.3 yields that for all λ:

$$p\left(A_{t+1} \leq A_t\left(1+\frac{s}{2}-\lambda\right)|A_t\right) \leq e^{-\frac{\lambda^2}{8}A_t}. \tag{6.C.5}$$

We can then bound this expression by using Equation 6.C.4 with $\lambda = \frac{s}{2}$, leading to

$$p(A_{t+1} < k | A_t \geq k) \leq e^{-\frac{s^2}{32}k} \text{ if } \alpha \leq \alpha_s.$$

Denoting by $N_{k,j}$ the number of messages sent between rounds k and j, we can decompose over $C\alpha^{-1}\log n$ rounds so that if m is such that there are at least α active nodes at round m then:

$$p(N_{m,m+C\alpha^{-1}\log n} \geq Cn\log n) \geq (1-e^{-\frac{s^2\alpha n}{32}})^{C\alpha^{-1}\log n},$$

because it is equal to the probability that the fraction of active nodes never goes below α for $C\alpha^{-1}\log n$ rounds. Therefore, if

$$s^2 \geq \frac{32}{\alpha n}\log\frac{C\log n}{\alpha\log(1-\delta)}, \text{ then } p(N_{m,m+C\alpha^{-1}\log n} \geq Cn\log n) \geq 1-\delta. \tag{6.C.6}$$

Equation 6.C.6 gives a lower bound on the value of α. Note that for a fixed α, this lower bound goes to 0 as n grows so in particular, Equation 6.C.6 is satisfied for $\alpha = \alpha_s$ if n is

large enough. It now remains to show that such a fraction α of active nodes can be reached in logarithmic time. Usual gossip analysis takes advantage of the exponential growth of the informed nodes during early rounds for which no collision occur. We have to adapt the analysis to the fact that nodes may stop communicating with some probability and split the analysis into two phases.

In the rest of the proof, we prove that a constant fraction of the nodes (independent of n) can be reached with a logarithmic number of rounds. We first analyze how long it takes to go from $\mathcal{O}(\log n)$ to $\mathcal{O}(n)$ active nodes and then from 1 to $\mathcal{O}(\log n)$. Equation 6.C.3 along with Equation 6.C.4 with $\lambda = \frac{s}{4}$ give that as long as $A_{t_0}(1+\frac{s}{4})^t \leq \alpha_s n$ then

$$p\left(A_{t+t_0+1} \geq A_{t_0}(1+\frac{s}{4})^{t+1} | A_t = A_{t_0}(1+\frac{s}{4})^t\right) \geq 1 - e^{-\frac{\alpha n s^2}{128}}$$

for any $t \geq t_0$ such that $A_{t_0}\left(1+\frac{c}{2}\right)^t \leq n\alpha_s$. Therefore, if we do this for all $t \leq t_{\alpha_s} = \frac{\log(\alpha_s n)}{\log(1+\frac{s}{4})}$ rounds (so for a logarithmic number of rounds) then $p\left(A_{t_{\alpha_s}+t_0} \geq n\alpha | A_{t_0}\right) \geq \left(1-e^{-\frac{A_{t_0}s^2}{128}}\right)^{t_{\alpha_s}}$ because in this case, $A_t \geq A_{t_0}$ for $t \geq t_0$. Therefore, if

$$A_{t_0} \geq -\frac{128}{s^2}\log\left(1-(1-\delta)^{\frac{1}{t_{\alpha_s}}}\right), \text{ then } p(A_{t_{\alpha_s}+t_0} \geq n\alpha_s | A_{t_0}) \geq 1-\delta. \tag{6.C.7}$$

Using the fact that $(1-x)^y \leq e^{-xy} \leq 1 - xy + \frac{x^2y^2}{2}$ along with the fact that $\delta < 1 \leq t_{\alpha_s}$ to simplify Equation 6.C.7, we show that if A_{t_0} satisfies:

$$A_{t_0} \geq \frac{128}{s^2}\log\left(\frac{2t_{\alpha_s}}{\delta}\right), \tag{6.C.8}$$

then it also satisfies Equation 6.C.7.

It only remains to prove that such an A_{t_0} can be reached with t_0 logarithmic in n. For this, use again Azuma inequality but on κ consecutive rounds this time. Therefore, Equation 6.C.5 becomes, assuming that at least A_t messages are sent at each round:

$$p\left(A_{t+\kappa} \leq A_t\left(1+\frac{\kappa s}{2}-\lambda\right)|A_t\right) \leq e^{-\frac{\lambda^2}{8\kappa}A_t}. \tag{6.C.9}$$

We apply this inequality for $\kappa = 2C\log(n)/s$ and $\lambda = C\log(n)/2$ (which is valid because at least $A_0 = 1$ node is active at each round), which yields:

$$p\left(A_\kappa \leq 1 + \frac{C\log(n)}{2}|A_t\right) \leq e^{-\frac{s\log(n)}{64}C}. \tag{6.C.10}$$

In particular, for a fixed values of C, s, and δ, then $p\left(A_\kappa \geq \frac{C\log(n)}{2}|A_t\right) \leq 1-\delta$ for n large enough. Finally, t_{α_s} is logarithmic in n so similarly, Equation (6.C.8) is satisfied for $t_0 = \kappa$ if n is large enough.

We conclude the proof by noting that

$$\begin{aligned}
&p\left(N_{0,t_0+t_{\alpha_s}+C\alpha^{-1}\log n} \geq Cn\log n\right) \\
&\geq p\left(A_{t_0} \geq \frac{128}{s^2}\log\left(\frac{2t_{\alpha_s}}{\delta}\right)\right) p\left(A_{t_{\alpha_s}+t_0} \geq n\alpha_s | A_{t_0} \geq \frac{128}{s^2}\log\left(\frac{2t_{\alpha_s}}{\delta}\right)\right) \\
&\qquad \times p\left(N_{t_{\alpha_s}+t_0, t_{\alpha_s}+t_0+C\alpha^{-1}\log n} \geq Cn\log n | A_{t_0+t_{\alpha_s}} \geq n\alpha_s\right) \\
&\geq (1-\delta)^3 \geq 1-3\delta.
\end{aligned}$$

Finally, we have that $t_0 \leq 2C\log(n)/s$, $t_{\alpha_s} \leq \log(n)/s$ and $1/\alpha_s \leq 3/s$ so in the end, $t_0 + t_{\alpha_s} + C\alpha^{-1}\log n \leq 6C\log(n)/s$. Without loss of generality, δ can also be replaced by $\delta/3$. □

REMARK 6 (Extension to the Asynchronous Version). *The first part of the proof directly extends to the asynchronous algorithm by simply considering slices of time during which a set of αn nodes send αn messages, which essentially means constant time. Then, we consider a logarithmic number of slices. The phase from 1 to $\mathcal{O}(\log n)$ active nodes requires sending a logarithmic number of messages and can thus be done in logarithmic time. Finally, phase 2 (going from $\mathcal{O}(\log n)$ to $\mathcal{O}(n)$ active nodes) consists in evaluating a logarithmic number of rounds during which a logarithmic number of nodes are active. Again, the only important thing is the number of messages sent (and not which node sent them) so using constant time intervals ensures that enough messages are sent between each pseudo-rounds with high probability. To summarize, it is possible to prove a statement very similar to that of Theorem 38 in the asynchronous setting, where the notion of rounds is replaced by constant time intervals. We omit the exact details of this alternative formulation.*

6.C.3. Maximum Likelihood Estimation.

PROOF OF THEOREM 39. We recall that S is the output sequence observed by curious nodes, so that S_0, is the first node that communicates with a curious node. The source is noted I_0, as it is the first node informed of the rumor. The set P is such that $p(I_0 = i) = 0$ if $i \notin P$. Recall that t_c is such that $S_{t_c} \in P$ and $S_t \notin P$ for $0 \leq t < t_c$. By a slight abuse of notation, note A_t the set of active nodes at the time where S_t is disclosed (time of t-th communication with a curious node), so $S_t \in A_t$ for all t.

We know that for all $i \in P$ then $p((S_t)_{t<t_c}|A_{t_c}, I_0 = i) = p((S_t)_{t<t_c}|A_{t_c})$ since $S_t \notin P$ for $t < t_c$. Similarly, $p((S_t)_{t\geq t_c}|A_{t_c}, I_0 = i, (S_t)_{t<t_c}) = p((S_t)_{t\geq t_c}|A_{t_c})$ since the output after some time only depends on the active nodes at that time. Therefore, $p(S|A_{t_c}, I_0 = i) = p(S|A_{t_c})$ for all $i \in P$, which critically relies on the fact that t_c is the time of first disclosure of a node in P (the first inequality would not hold otherwise). We note $[n] = \{1, ..., n\}$. We then write for $i \in P$:

$$\begin{aligned} p(I_0 = i|S) &= \sum_{A\subset[n]} p(A_{t_c} = A|S)p(I_0 = i|A_{t_c} = A, S) \\ &= \sum_{A\subset[n]} p(A_{t_c} = A|S)p(I_0 = i|A_{t_c} = A) \\ &= \sum_{A\subset[n]:S_{t_c}\in A} p(A_{t_c} = A|S)\frac{p(I_0 = i)}{p(A_{t_c} = A)}p(A_{t_c} = A|I_0 = i). \end{aligned}$$

Let $j \in P \cap A_{t_c}$. If $i \in A_{t_c}$ then $p(A_{t_c} = A|I_0 = i) = p(A_{t_c} = A|I_0 = j)$. Otherwise, let us denote $E_{ij}(A) = \cap_{k\in A\setminus\{j\}}\{\text{k active at time } t_c\} \cap_{k\notin A\cup\{i\}} \{\text{k inactive at time } t_c\}$. This event represents the realization of A_{t_c} for all nodes different from i and j. We then write:

$$\begin{aligned} p(A_{t_c} = A|I_0 = i) &= p(\cap_{k\in A}\{k \in A_{t_c}\} \cap_{k\notin A} \{k \notin A_{t_c}\}|I_0 = i) \\ &= p(E_{ij}(A)|I_0 = i)p(j \in A_{t_c}, i \notin A_{t_c}|I_0 = i, E_{ij}(A)) \\ &= p(E_{ij}(A)|I_0 = j)p(j \in A_{t_c}, i \notin A_{t_c}|I_0 = i, E_{ij}(A)) \\ &\leq p(E_{ij}(A)|I_0 = j)p(j \in A_{t_c}, i \notin A_{t_c}|I_0 = j, E_{ij}(A))) \end{aligned}$$

$$= p(A_{t_c} = A | I_0 = j).$$

The inequality comes from the fact that it is more likely that j is active and i is inactive if j is the source than if i is (i.e., if it is already the case at the beginning). This means that $p(A_{t_c} = A | I_0 = i) \leq p(A_{t_c} = A | I_0 = j)$ for all $i \in [n]$ and $j \in A_{t_c}$. Since the summation is over all A such that $S_0 \in A$ (by definition of S_{t_c} and A_{t_c}), and $p(A_{t_c} = A | I_0 = i) \leq p(A_{t_c} = A | I_0 = S_{t_c})$, we have for all considered A:

$$\begin{aligned} p(I_0 = i | S) &= \sum_{A \subset [n] : S_{t_c} \in A} p(A_{t_c} = A | S) \frac{p(I_0 = i)}{p(A_{t_c} = A)} p(A_{t_c} = A | I_0 = i) \\ &\leq \sum_{A \subset [n] : S_{t_c} \in A} p(A_{t_c} = A | S) \frac{p(I_0 = S_{t_c})}{p(A_{t_c} = A)} p(A_{t_c} = A | I_0 = S_{t_c}) \\ &= p(I_0 = S_{t_c} | S). \end{aligned}$$

This means that S_{t_c} is more likely to be the source than any other suspected node when the adversary observes output S. Note that this requires uniform prior over nodes that can be suspected since we used the fact that $p(I_0 = i) = p(I_0 = S_{t_c})$ for all $i \in P$. For $i \notin P$, $p(I_0 = i | S) = 0 \leq p(I_0 = S_{t_c} | S)$. □

6.D. Challenges of Private Gossip for General Graphs

A natural extension of this work is to consider general graphs. We discuss in this section several aspects related to the natural privacy of gossip protocols in arbitrary graphs. In particular, we highlight the fact that problem-specific modeling choices are needed to go beyond the complete graph, and that even defining a notion of privacy that is suitable for all graphs is very challenging.

6.D.1. Average-Case versus Worst-Case Privacy. Unlike the case of complete graphs, the location of curious nodes critically impacts the privacy guarantees in arbitrary graphs. A naive way to deal with this issue is to randomize the location of curious nodes *a posteriori*. Let us denote by $\mathcal{L}_{i,j}^f$ the set containing all subsets of nodes of size f of the graph that do not contain i and j. For fixed nodes i and j, the set of curious nodes $\mathcal{C}$ is drawn from $U(\mathcal{L}_{i,j}^f)$, the uniform distribution over $\mathcal{L}_{i,j}^f$. For some parameters $\epsilon, \delta \geq 0$, privacy can be defined as follows: $\forall i, j \in \{0, ..., n-1\}$, $\forall S \in \mathcal{S}$

$$\mathbb{E}_{\mathcal{C} \sim U(\mathcal{L}_{i,j}^f)} [p_i(S, \mathcal{C}) - e^{\epsilon} p_j(S, \mathcal{C})] \leq \delta.$$

Note that $p_i(S, \mathcal{C}) = 0$ if the output sequence S is not compatible with the set of curious nodes $\mathcal{C}$, i.e. if $(k, l) \in S$ and $k, l \notin \mathcal{C}$. To pick the curious nodes, it is possible to either pick a set of f curious nodes at once or to pick each node (except for i and j) with probability f/n. This randomized definition allows to prove a bound similar to that of Theorem 35 for arbitrary graphs. Indeed, the first node that receives the rumor has probability $\frac{f}{n}$ of being a curious node. However, such average-case notions of privacy are highly undesirable: in this case, no protection is provided against a (much more realistic) adversary that controls a fraction of nodes fixed in advance.

The worst-case approach consists in bounding the maximum difference instead of the expectation. This is the approach taken in our work for the complete graph (the max

operator is implicit because the location of curious nodes does not matter in a complete graph). In the case of general graphs, the corresponding privacy definition is given by: $\forall i, j \in \{0, ..., n-1\},\ \forall S \in \mathcal{S}$,

$$\max_{\mathcal{C} \in \mathcal{L}^f_{i,j}} [p_i(S, \mathcal{C}) - e^{\epsilon} p_j(S, \mathcal{C})] \leq \delta.$$

We immediately observe that with this definition, it is impossible to have $\delta < 1$ as soon as there is a node in the graph with less than f neighbors. This modeling choice is quite unrealistic as well because having a node surrounded by curious nodes means that the adversary actually believes this specific node has a strong probability of being the source and therefore put more sensors around it. A possible alternative would be to place curious nodes so as to bound the maximum privacy for any pair of nodes, and then evaluate the minimum privacy in this setting. This definition would mean that the adversary wants to be able to distinguish any pair of nodes as best as possible.

We see that choosing the locations of the curious nodes in an arbitrary graph is a complex problem that is heavily dependent on the topology of the graph and on the prior of the adversary on the locations of the curious nodes. Indeed, the adversary may simply want to isolate a sufficiently small group of nodes that have a high probability of being the source.

6.D.2. Relaxing the Differential Privacy Definition. Differential privacy is a very strong notion that enforces indistinguishability between all pairs of nodes, in order to be robust to any prior information about who might be the source. In particular, an adversary should not be able to precisely identify the source even if it knows that only two nodes in the graph can be the source. Although it was possible to obtain meaningful privacy guarantees of this kind for the complete graph, this appears to be too strong of a requirement for some graph topology and location of curious nodes. Consider for instance the extreme case of a line graph. It is clear that any non-trivial adversary can always distinguish between two segments of the line. This intuition directly extends to any graph which admits a cut with only curious nodes in it.

A natural idea is to restrict the pairs of nodes that are required to be indistinguishable. Several ways of doing this may be considered. For instance, one could require that each node is indistinguishable from k other nodes in the graph. Such relaxed definition could be obtained using the Pufferfish framework Kifer and Machanavajjhala [2014], which explicitly provides a notion of secret to protect. But how to choose such k nodes based on the topology and how to characterize the optimal locations of curious nodes is very challenging. Another direction could be to adapt the notions of metric-based differential privacy Andrés et al. [2013], Chatzikokolakis et al. [2013] to design a notion of privacy where the required indistinguishability for a given node is a function of its distance to curious nodes in the graph, or to require that pairs of nodes become less indistinguishable with distance in the graph. Yet, it is not clear how to characterize the influence of the graph topology.

6.D.3. Optimality of Algorithm 11 with s=0. We have seen in this section that the privacy guarantees for arbitrary graphs heavily rely on the particular privacy notion and that some recent privacy frameworks may provide tools to relax the classic differential privacy definition which is generally too strong for arbitrary graphs. We conjecture that for some of these relaxed definitions, the optimal algorithm for general graphs will be the same as in our case of the complete graph. Indeed, the strength of Algorithm 11 with $s = 0$ is

to forget initial conditions quickly. In the complete graph, it does so in one step. In an arbitrary graph, the information about the part of the graph the source belongs to is still present after some steps, but the source should quickly be completely indistinguishable from its direct neighbors. In particular, attacks based on centrality Shah and Zaman [2011] are rather meaningless against this algorithm because spreading only occurs along a random walk in the graph. As in the case of the complete graph, Algorithm 11 with $s > 0$ is then likely to enjoy near-optimal privacy guarantees.

Conclusion and Research Directions

Summary of the thesis

In this thesis, we have presented several algorithms that extend the recent advances in convex optimization to the distributed setting, in particular in the case of decentralized optimization.

In the introduction, we have presented the basic and more recent results in optimization for machine learning, and in particular acceleration and variance-reduced methods for finite sums. Then, we have given a brief overview of the distributed optimization models and literature, with a focus on the dual approach for decentralized optimization.

Our first contribution can be split into 3 major parts, that are represented by Chapters 2, 3 and 4. The first part builds on the results of Scaman et al. [2017b] and leverages the dual approach to design an accelerated decentralized algorithm that uses randomized pairwise communications instead of global synchronous ones. In particular, as a special case of our algorithm, we obtain the first direct acceleration of randomized gossip that is not tailored to specific graphs. The second part of this contribution focuses on the problem in which local functions f_i are finite sums as well. In this case, we show that using an alternative dual formulation that has an augmented graph interpretation allows to have an accelerated variance-reduced algorithm with separated communication and computation steps. We also give a lower bound and prove that the synchronous version of this algorithm is optimal. Finally, we show that the same framework can be used to obtain primal algorithms by using a dual-free trick that relies on the use of Bregman divergences.

The second contribution of this thesis focuses on reducing the communication cost in the centralized setting. To do so, we study a statistical preconditioning algorithm similar to DANE [Shamir et al., 2014], in which the server uses a local dataset which is small compared to the global one in order to make more efficient updates. Then, we leverage an accelerated version of the Bregman gradients algorithm to show that the convergence rate of this algorithm depends on the relative condition number between the objective function on the server's dataset and on the global one, and prove improved dependency on this quantity thanks to acceleration. Finally, we analyze stochastic (variance-reduced) algorithms that deal with the case in which all nodes cannot participate in every iteration, which is very standard in federated learning.

In the last contribution, we do not study optimization directly, but the spreading of sensitive information in a graph. In machine learning, this problem comes up if a one node owns sensitive data that from which it is possible to infer whether a given model has been trained on it or not. In our case, the node agrees to use for training, but would not like curious nodes to be able to trace this data back to him. This happens for instance if only one node has non-zero entries for some dimensions. In this case, we precisely define and quantify

the privacy guarantees for the node that spreads the sensitive information, and give protocols that have near-optimal privacy guarantees while also having order-optimal spreading speed.

Perspectives

As stated in the summary, this thesis extends several convex optimization result to the distributed setting, and in particular in the decentralized case. This raises new questions, and some of the natural extensions are:

(1) Scaman et al. [2017b] proved that the usual lower bounds for convex optimization extended to the decentralized setting with an added term to account for communications, and give an optimal algorithm. In this thesis, we further extend these results (lower bound and optimal algorithm) to the pairwise gossip and finite sum cases. Yet, all these were obtained using a dual approach. Kovalev et al. [2020] obtain an optimal algorithm that relies on primal oracles only, and Li et al. [2020c] get the same result for the finite sum setting, although they rely on mini-batching stochastic computations to achieve optimality. Obtaining primal optimal decentralized algorithms that use randomized gossip communications and do not rely on batching computations is an interesting problem that remains open.

(2) The accelerated randomized gossip algorithm obtained in Chapter 2 requires a global counter of updates in order to perform the convex combinations required for acceleration, which implies some level of coordination between agents. It is actually possible to bypass this limitation by studying the algorithm in continuous time, and making continuous time contractions but discrete time updates, as shown in *Asynchrony and Acceleration in Gossip Algorithms* [Even, Hendrikx, and Massoulié, 2020]. Leveraging this continuous framework to lift this coordination for finite sum algorithms would allow to apply them using a synchronized clock only.

(3) Although many of the algorithms presented in this thesis rely on randomized gossip updates and are thus not synchronous, their analysis does not extend to the case of delayed updates, which is how asynchrony is often understood. It is actually possible to consider randomized gossip algorithms with delays, as shown in *Decentralized Optimization with Heterogeneous Delays: a Continuous-Time Approach* [Even, Hendrikx, and Massoulie, 2021]. Considering (accelerated) variance-reduced algorithms with delays (and possibly primal updates) would allow to design very realistic and practical algorithms for modern peer-to-peer networks.

(4) Statistical preconditioning through the use of Bregman gradient algorithms allows to drastically reduce the communication cost of training distributed machine learning models, and shows great practical performance. Yet, and although it is not what is observed in practice, our convergence Theorems only show asymptotic acceleration (which is consistent with the impossibility result from Dragomir et al. [2021b]). A better understanding of this discrepancy between the theory and the practice, and when non-asymptotic acceleration can actually be achieved is a key open question in the theory of statistical preconditionning and Bregman methods in general.

(5) Although there are interplays with statistics through preconditioning, this thesis focuses mainly on optimizing the training error. It would be interesting to study the test error in a distributed setting, for instance when each node samples data from its own distribution, as in Richards and Rebeschini [2020].

(6) Chapter 6 introduces a framework to analyze privacy issues when sharing information (or when collaboratively training a model), and shows how to design algorithm that achieve a good privacy-speed tradeoff (order-optimal in both). Yet, this analysis only applies to complete communication graphs, and extending the definitions and results to more complex graphs is a very challenging question that we discuss, but unfortunately leave open.

Bibliography

Martín Abadi, Andy Chu, Ian J. Goodfellow, H. Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. Deep learning with differential privacy. In *CCS*, 2016.

Huseyin Acan, Andrea Collevecchio, Abbas Mehrabian, and Nick Wormald. On the push&pull protocol for rumor spreading. *SIAM Journal on Discrete Mathematics*, 31 (2):647–668, 2017.

Sulaiman A Alghunaim, Kun Yuan, and Ali H Sayed. A linearly convergent proximal gradient algorithm for decentralized optimization. *arXiv preprint arXiv:1905.07996*, 2019.

Dan Alistarh, Seth Gilbert, Rachid Guerraoui, and Morteza Zadimoghaddam. How efficient can gossip be?(on the cost of resilient information exchange). In *ICALP*, 2010.

Zeyuan Allen-Zhu. Katyusha: The first direct acceleration of stochastic gradient methods. In *Proceedings of Symposium on Theory of Computing*, pages 1200–1205, 2017.

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*, 2016.

Noga Alon and Vitali D Milman. $\lambda 1$, isoperimetric inequalities for graphs, and superconcentrators. *Journal of Combinatorial Theory, Series B*, 38(1):73–88, 1985.

Miguel E. Andrés, Nicolás Emilio Bordenabe, Konstantinos Chatzikokolakis, and Catuscia Palamidessi. Geo-indistinguishability: differential privacy for location-based systems. In *CCS*, 2013.

Dana Angluin, James Aspnes, and David Eisenstat. Fast computation by population protocols with a leader. *Distributed Computing*, 21(3):183–199, 2008.

Yossi Arjevani and Ohad Shamir. Communication complexity of distributed convex learning and optimization. In *Advances in Neural Information Processing Systems 28*, pages 1756–1764, 2015.

Yossi Arjevani, Joan Bruna, Bugra Can, Mert Gürbüzbalaban, Stefanie Jegelka, and Hongzhou Lin. Ideal: Inexact decentralized accelerated augmented lagrangian method. *arXiv preprint arXiv:2006.06733*, 2020.

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

Mahmoud Assran, Nicolas Loizou, Nicolas Ballas, and Mike Rabbat. Stochastic gradient push for distributed deep learning. In *International Conference on Machine Learning*, pages 344–353. PMLR, 2019.

Mahmoud Assran, Arda Aytekin, Hamid Reza Feyzmahdavian, Mikael Johansson, and Michael G Rabbat. Advances in asynchronous parallel and distributed optimization. *Proceedings of the IEEE*, 108(11), 2020.

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.

Francis Bach and Eric Moulines. Non-asymptotic analysis of stochastic approximation algorithms for machine learning. *Advances in neural information processing systems*, 24: 451–459, 2011a.

Francis Bach and Eric Moulines. Non-asymptotic analysis of stochastic approximation algorithms for machine learning. In *Advances in neural information processing systems*, 2011b.

Borja Balle, James Bell, Adria Gascon, and Kobbi Nissim. The Privacy Blanket of the Shuffle Model. Technical report, arXiv:1903.02837, 2019.

Heinz H Bauschke, Jérôme Bolte, and Marc Teboulle. A descent lemma beyond lipschitz gradient continuity: first-order methods revisited and applications. *Mathematics of Operations Research*, 42(2):330–348, 2017.

Amir Beck. *First-Order Methods in Optimization*. MOS-SIAM Series on Optimization. SIAM, 2017.

Amir Beck and Marc Teboulle. Mirror descent and nonlinear projected subgradient methods for convex optimization. *Operations Research Letters*, 31(3):167–175, 2003.

Amir Beck and Marc Teboulle. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM Journal on Imaging Sciences*, 2(1):183–202, 2009a.

Amir Beck and Marc Teboulle. A fast iterative shrinkage-thresholding algorithm with application to wavelet-based image deblurring. In *2009 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 693–696. IEEE, 2009b.

Aurélien Bellet, Rachid Guerraoui, Mahsa Taziki, and Marc Tommasi. Personalized and Private Peer-to-Peer Machine Learning. In *AISTATS*, 2018.

Aurélien Bellet, Rachid Guerraoui, and Hadrien Hendrikx. Who started this rumor? quantifying the natural differential privacy of gossip protocols. In *34th International Symposium on Distributed Computing (DISC 2020)*. Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2020.

Petra Berenbrink, Jurek Czyzowicz, Robert Elsässer, and Leszek Gasieniec. Efficient information exchange in the random phone-call model. *Automata, Languages and Programming*, 2010.

Raphaël Berthier, Francis Bach, and Pierre Gaillard. Accelerated gossip in networks of given dimension using jacobi polynomial iterations. *SIAM Journal on Mathematics of Data Science*, 2(1):24–47, 2020.

Pascal Bianchi, Walid Hachem, and Franck Iutzeler. A coordinate descent primal-dual algorithm and application to distributed asynchronous optimization. *IEEE Transactions on Automatic Control*, 61(10):2947–2957, 2015.

Béla Bollobás. *Random graphs*. Number 73 in Cambridge studies in advanced mathematics. Cambridge University Press, 2001.

Léon Bottou. Large-scale machine learning with stochastic gradient descent. In *Proceedings of COMPSTAT*, pages 177–186. Springer, 2010.

Léon Bottou, Frank E Curtis, and Jorge Nocedal. Optimization methods for large-scale machine learning. *Siam Review*, 60(2):223–311, 2018.

Stéphane Boucheron, Gábor Lugosi, and Pascal Massart. *Concentration Inequalities: A Nonasymptotic Theory of Independence*. Oxford University Press, 2013.

Stephen Boyd, Stephen P Boyd, and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

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, and Jonathan Eckstein. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends in Machine Learning*, 3(1):1–122, 2010.

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.

Sébastien Bubeck. Introduction to online optimization. *Lecture Notes*, 2, 2011.

Sébastien Bubeck. Convex optimization: Algorithms and complexity. *Foundations and Trends® in Machine Learning*, 8(3-4):231–357, 2015.

Loris Cannelli, Francisco Facchinei, Gesualdo Scutari, and Vyacheslav Kungurtsev. Asynchronous optimization over graphs: Linear convergence under error bound conditions. *IEEE Transactions on Automatic Control*, 2020.

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.

Antonin Chambolle and Charles Dossal. On the convergence of the iterates of" fista". *Journal of Optimization Theory and Applications*, 166(3):25, 2015.

Antonin Chambolle and Thomas Pock. An introduction to continuous optimization for imaging. *Acta Numerica*, 25:161–319, 2016.

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

Konstantinos Chatzikokolakis, Miguel E. Andrés, Nicolás Emilio Bordenabe, and Catuscia Palamidessi. Broadening the Scope of Differential Privacy Using Metrics. In *PETS*, 2013.

Kamalika Chaudhuri, Claire Monteleoni, and Anand D. Sarwate. Differentially Private Empirical Risk Minimization. *Journal of Machine Learning Research*, 12:1069–1109, 2011.

Gong Chen and Marc Teboulle. Convergence analysis of a proximal-like minimization algorithm using Bregman functions. *SIAM Journal on Optimization*, 3(3):538–543, August 1993.

Jianmin Chen, Xinghao Pan, Rajat Monga, Samy Bengio, and Rafal Jozefowicz. Revisiting distributed synchronous SGD. *arXiv preprint arXiv:1604.00981*, 2016.

Albert Cheu, Adam D. Smith, Jonathan Ullman, David Zeber, and Maxim Zhilyaev. Distributed Differential Privacy via Shuffling. Technical report, arXiv:1808.01394, 2018.

Yun Kuen Cheung, Richard Cole, and Yixin Tao. Fully asynchronous stochastic coordinate descent: a tight lower bound on the parallelism achieving linear speedup. *Mathematical Programming*, pages 1–63, 2020.

Yun Kuen Cheung, Richard J Cole, and Yixin Tao. Parallel stochastic asynchronous coordinate descent: Tight bounds on the possible parallelism. *SIAM Journal on Optimization*, 31(1):448–460, 2021.

Igor Colin, Aurélien Bellet, Joseph Salmon, and Stéphan Clémençon. Gossip dual averaging for decentralized optimization of pairwise functions. In *Proceedings of the International Conference on International Conference on Machine Learning-Volume 48*, pages 1388–1396, 2016.

Patrick L Combettes and Jean-Christophe Pesquet. Proximal splitting methods in signal processing. In *Fixed-point algorithms for inverse problems in science and engineering*, pages 185–212. Springer, 2011.

Alexandre d'Aspremont, Damien Scieur, and Adrien Taylor. Acceleration methods. *arXiv preprint arXiv:2101.09545*, 2021.

Peter Davies, Vijaykrishna Gurunathan, Niusha Moshrefi, Saleh Ashkboos, and Dan Alistarh. Distributed variance reduction with optimal communication. *arXiv preprint arXiv:2002.09268*, 2020.

Aaron Defazio. A simple practical accelerated method for finite sums. In *Advances in Neural Information Processing Systems*, pages 676–684, 2016.

Aaron Defazio, Francis Bach, and Simon Lacoste-Julien. SAGA: A fast incremental gradient method with support for non-strongly convex composite objectives. In *Advances in Neural Information Processing Systems*, pages 1646–1654, 2014a.

Aaron Defazio, Justin Domke, et al. Finito: A faster, permutable incremental gradient method for big data problems. In *International Conference on Machine Learning*, pages 1125–1133. PMLR, 2014b.

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

Alan Demers, Dan Greene, Carl Hauser, Wes Irish, John Larson, Scott Shenker, Howard Sturgis, Dan Swinehart, and Doug Terry. Epidemic algorithms for replicated database maintenance. In *PODC*, 1987.

Paolo Di Lorenzo and Gesualdo Scutari. Distributed nonconvex optimization over networks. In *2015 IEEE 6th International Workshop on Computational Advances in Multi-Sensor Adaptive Processing (CAMSAP)*, pages 229–232. IEEE, 2015.

Paolo Di Lorenzo and Gesualdo Scutari. Next: In-network nonconvex optimization. *IEEE Transactions on Signal and Information Processing over Networks*, 2(2):120–136, 2016.

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.

Aymeric Dieuleveut, Nicolas Flammarion, and Francis Bach. Harder, better, faster, stronger convergence rates for least-squares regression. *The Journal of Machine Learning Research*, 18(1):3520–3570, 2017.

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.

Benjamin Doerr, Mahmoud Fouz, and Tobias Friedrich. Social networks spread rumors in sublogarithmic time. In *STOC*, 2011.

Radu-Alexandru Dragomir, Adrien Taylor, Alexandre d'Aspremont, and Jérôme Bolte. Optimal complexity and certification of Bregman first-order methods. *arXiv preprint arXiv:1911.08510*, 2019.

Radu-Alexandru Dragomir, Mathieu Even, and Hadrien Hendrikx. Fast stochastic bregman gradient methods: Sharp analysis and variance reduction. *International Conference on Machine Learning*, 2021a.

Radu-Alexandru Dragomir, Adrien B Taylor, Alexandre d'Aspremont, and Jérôme Bolte. Optimal complexity and certification of bregman first-order methods. *Mathematical Programming*, pages 1–43, 2021b.

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, 2012a.

John C Duchi, Alekh Agarwal, and Martin J Wainwright. Dual averaging for distributed optimization: Convergence analysis and network scaling. *IEEE Transactions on Automatic Control*, 2012b.

Darina Dvinskikh and Alexander Gasnikov. Decentralized and parallelized primal and dual accelerated methods for stochastic convex programming problems. *arXiv preprint arXiv:1904.09015*, 2019.

Darina Dvinskikh and Alexander Gasnikov. Decentralized and parallel primal and dual accelerated methods for stochastic convex programming problems. *Journal of Inverse and Ill-posed Problems*, 2021.

Cynthia Dwork. Differential Privacy. In *Proceedings of the 33rd International Colloquium on Automata, Languages and Programming*, pages 1–12, 2006.

Cynthia Dwork and Aaron Roth. The algorithmic foundations of differential privacy. *Foundations and Trends in Theoretical Computer Science*, 9(3–4):211–407, 2014.

Cynthia Dwork, Krishnaram Kenthapadi, Frank McSherry, Ilya Mironov, and Moni Naor. Our data, ourselves: Privacy via distributed noise generation. In *EUROCRYPT*, 2006.

Paul Erdős and Alfred Rényi. On a classical problem of probability theory. *Publ. Math. Inst. Hung. Acad. Sci.*, 1961.

Úlfar Erlingsson, Vitaly Feldman, Ilya Mironov, and Ananth Raghunathan abd Kunal Talwar. Amplification by Shuffling: From Local to Central Differential Privacy via Anonymity. Technical report, arXiv:1811.12469, 2018.

Patrick T Eugster, Rachid Guerraoui, Anne-Marie Kermarrec, and Laurent Massoulié. Epidemic information dissemination in distributed systems. *Computer*, 37(5):60–67, 2004.

Mathieu Even, Hadrien Hendrikx, and Laurent Massoulié. Asynchrony and acceleration in gossip algorithms. *arXiv preprint arXiv:2011.02379*, 2020.

Mathieu Even, Hadrien Hendrikx, and Laurent Massoulie. Decentralized optimization with heterogeneous delays: a continuous-time approach. *arXiv preprint arXiv:2106.03585*, 2021.

Eyal Even-Dar, Sham M Kakade, and Yishay Mansour. Online markov decision processes. *Mathematics of Operations Research*, 34(3):726–736, 2009.

Giulia Fanti, Peter Kairouz, Sewoong Oh, Kannan Ramchandran, and Pramod Viswanath. Hiding the rumor source. *IEEE Transactions on Information Theory*, 2017.

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

Alan M Frieze and Geoffrey R Grimmett. The shortest-path problem for graphs with random arc-lengths. *Discrete Applied Mathematics*, 10(1):57–77, 1985.

Chryssis Georgiou, Seth Gilbert, and Dariusz R Kowalski. Confidential gossip. In *ICDCS*, 2011.

Chryssis Georgiou, Seth Gilbert, Rachid Guerraoui, and Dariusz R Kowalski. Asynchronous gossip. *Journal of the ACM*, 60(2):11, 2013.

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

Mohsen Ghaffari and Calvin Newport. How to discreetly spread a rumor in a crowd. In *DISC*, 2016.

George Giakkoupis, Rachid Guerraoui, Arnaud Jégou, Anne-Marie Kermarrec, and Nupur Mittal. Privacy-conscious information diffusion in social networks. In *International Symposium on Distributed Computing*, pages 480–496. Springer, 2015.

George Giakkoupis, Yasamin Nazari, and Philipp Woelfel. How asynchrony affects rumor spreading time. In *PODC*, 2016.

Eduard Gorbunov, Alexander Rogozin, Aleksandr Beznosikov, Darina Dvinskikh, and Alexander Gasnikov. Recent theoretical advances in decentralized distributed convex optimization. *arXiv preprint arXiv:2011.13259*, 2020.

Eduard Gorbunov, Filip Hanzely, and Peter Richtárik. Local sgd: Unified theory and new efficient methods. In *International Conference on Artificial Intelligence and Statistics*, pages 3556–3564. PMLR, 2021.

Karol Gotfryd, Marek Klonowski, and Dominik Pajak. On location hiding in distributed systems. In *International Colloquium on Structural Information and Communication Complexity*, pages 174–192. Springer, 2017.

Robert M Gower, Filip Hanzely, Peter Richtárik, and Sebastian U. 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 Mansel Gower, Nicolas Loizou, Xun Qian, Alibek Sailanbayev, Egor Shulgin, and Peter Richtárik. Sgd: General analysis and improved rates. In *International Conference on Machine Learning*, pages 5200–5209. PMLR, 2019.

Priya Goyal, Piotr Dollár, Ross Girshick, Pieter Noordhuis, Lukasz Wesolowski, Aapo Kyrola, Andrew Tulloch, Yangqing Jia, and Kaiming He. Accurate, large minibatch SGD: Training imagenet in 1 hour. *arXiv preprint arXiv:1706.02677*, 2017.

Osman Güler. New proximal point algorithms for convex minimization. *SIAM Journal on Optimization*, 2(4):649–664, 1992.

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

Filip Hanzely and Peter Richtárik. Fastest rates for stochastic mirror descent methods. *arXiv preprint arXiv:1803.07374*, 2018.

Filip Hanzely and Peter Richtárik. Accelerated coordinate descent with arbitrary sampling and best rates for minibatches. In *Artificial Intelligence and Statistics*, 2019.

Filip Hanzely, Peter Richtarik, and Lin Xiao. Accelerated Bregman proximal gradient methods for relatively smooth convex optimization. arXiv:1808.03045, 2018.

Filip Hanzely, Peter Richtarik, and Lin Xiao. Accelerated bregman proximal gradient methods for relatively smooth convex optimization. *Computational Optimization and Applications*, pages 1–36, 2021.

Elad Hazan. The convex optimization approach to regret minimization. *Optimization for machine learning*, page 287, 2012.

Xi He, Rachael Tappenden, and Martin Takáč. Dual free adaptive minibatch sdca for empirical risk minimization. *Frontiers in Applied Mathematics and Statistics*, 4:33, 2018.

Hadrien Hendrikx, Francis Bach, and Laurent Massoulié. Accelerated decentralized optimization with local updates for smooth and strongly convex objectives. In *Artificial Intelligence and Statistics*, 2019a.

Hadrien Hendrikx, Francis Bach, and Laurent Massoulié. An accelerated decentralized stochastic proximal algorithm for finite sums. In *Advances in Neural Information Processing Systems*, 2019b.

Hadrien Hendrikx, Francis Bach, and Laurent Massoulié. Dual-free stochastic decentralized optimization with variance reduction. In *Advances in Neural Information Processing Systems*, 2020a.

Hadrien Hendrikx, Francis Bach, and Laurent Massoulie. An optimal algorithm for decentralized finite sum optimization. *arXiv preprint arXiv:2005.10675*, 2020b.

Hadrien Hendrikx, Lin Xiao, Sebastien Bubeck, Francis Bach, and Laurent Massoulie. Statistically preconditioned accelerated gradient method for distributed optimization. In *International Conference on Machine Learning*, pages 4203–4227. PMLR, 2020c.

Herbert W. Hethcote. The mathematics of infectious diseases. *SIAM Review*, 42(4):599–653, 2000.

Wassily Hoeffding. Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association*, 58(301):13–30, 1963.

Zhiyi Huang and Jinyan Liu. Optimal differentially private algorithms for k-means clustering. In *PODS*, 2018.

Franck Iutzeler, Pascal Bianchi, Philippe Ciblat, and Walid Hachem. Asynchronous distributed optimization using a randomized alternating direction method of multipliers. In *52nd IEEE conference on decision and control*, pages 3671–3676. IEEE, 2013.

Martin Jaggi, Virginia Smith, Martin Takac, Jonathan Terhorst, Sanjay Krishnan, Thomas Hofmann, and Michael I. Jordan. Communication-efficient distributed dual coordinate ascent. In *Advances in Neural Information Processing Systems 27*, pages 3068–3076, 2014.

Dušan Jakovetić. A unification and generalization of exact distributed first-order methods. *IEEE Transactions on Signal and Information Processing over Networks*, 5(1):31–46, 2018.

Dusan Jakovetic, Joao Xavier, and José MF Moura. Cooperative convex optimization in networked systems: Augmented lagrangian algorithms with directed gossip communication. *IEEE Transactions on Signal Processing*, 59(8):3889–3902, 2011.

Dušan Jakovetić, José MF Moura, and Joao Xavier. Linear convergence rate of a class of distributed augmented lagrangian algorithms. *IEEE Transactions on Automatic Control*, 60(4):922–936, 2014.

Jiaojiao Jiang, Sheng Wen, Shui Yu, Yang Xiang, and Wanlei Zhou. Identifying propagation sources in networks: State-of-the-art and comparative studies. *IEEE Communications Surveys & Tutorials*, 19(1):465–481, 2017.

Björn Johansson, Maben Rabi, and Mikael Johansson. A randomized incremental subgradient method for distributed optimization in networked systems. *SIAM Journal on Optimization*, (3), 2009.

Rie Johnson and Tong Zhang. Accelerating stochastic gradient descent using predictive variance reduction. In *Advances in Neural Information Processing Systems 26*, pages 315–323, 2013a.

Rie Johnson and Tong Zhang. Accelerating stochastic gradient descent using predictive variance reduction. In *Advances in Neural Information Processing Systems*, pages 315–323, 2013b.

Peter Kairouz, H Brendan McMahan, Brendan Avent, Aurélien Bellet, Mehdi Bennis, Arjun Nitin Bhagoji, Keith Bonawitz, Zachary Charles, Graham Cormode, Rachel Cummings, et al. Advances and open problems in federated learning. *arXiv preprint arXiv:1912.04977*, 2019.

Sham Kakade, Shai Shalev-Shwartz, Ambuj Tewari, et al. 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.

Sai Praneeth Karimireddy, Satyen Kale, Mehryar Mohri, Sashank Reddi, Sebastian Stich, and Ananda Theertha Suresh. Scaffold: Stochastic controlled averaging for federated learning. In *International Conference on Machine Learning*, pages 5132–5143. PMLR, 2020.

Richard Karp, Christian Schindelhauer, Scott Shenker, and Berthold Vocking. Randomized rumor spreading. In *FOCS*, 2000.

David Kempe, Alin Dobra, and Johannes Gehrke. Gossip-Based Computation of Aggregate Information. In *FOCS*, 2003a.

David Kempe, Alin Dobra, and Johannes Gehrke. Gossip-based computation of aggregate information. In *44th Annual IEEE Symposium on Foundations of Computer Science, 2003. Proceedings.*, pages 482–491. IEEE, 2003b.

Daniel Kifer and Ashwin Machanavajjhala. Pufferfish: A framework for mathematical privacy definitions. *ACM Transactions on Database Systems*, 39(1):3, 2014.

Anastasia Koloskova, Tao Lin, Sebastian U Stich, and Martin Jaggi. Decentralized deep learning with arbitrary communication compression. *arXiv preprint arXiv:1907.09356*, 2019a.

Anastasia Koloskova, Sebastian Stich, and Martin Jaggi. Decentralized Stochastic Optimization and Gossip Algorithms with Compressed Communication. In *ICML*, 2019b.

Anastasia Koloskova, Nicolas Loizou, Sadra Boreiri, Martin Jaggi, and Sebastian Stich. A unified theory of decentralized sgd with changing topology and local updates. In *International Conference on Machine Learning*, pages 5381–5393. PMLR, 2020.

Jakub Konečnỳ, H Brendan McMahan, Felix X Yu, Peter Richtárik, Ananda Theertha Suresh, and Dave Bacon. Federated learning: Strategies for improving communication efficiency. *arXiv preprint arXiv:1610.05492*, 2016.

Dmitry Kovalev, Adil Salim, and Peter Richtárik. Optimal and practical algorithms for smooth and strongly convex decentralized optimization. *Advances in Neural Information Processing Systems*, 2020.

Dmitry Kovalev, Anastasia Koloskova, Martin Jaggi, Peter Richtarik, and Sebastian Stich. A linearly convergent algorithm for decentralized optimization: Sending less bits for free! In *International Conference on Artificial Intelligence and Statistics*, pages 4087–4095. PMLR, 2021.

Dariusz R Kowalski and Christopher Thraves Caro. Estimating time complexity of rumor spreading in ad-hoc networks. In *International Conference on Ad-Hoc Networks and Wireless*, pages 245–256. Springer, 2013.

Guanghui Lan and Yi Zhou. An optimal randomized incremental gradient method. *Mathematical programming*, pages 1–49, 2017.

Rémi Leblond, Fabian Pedregosa, and Simon Lacoste-Julien. ASAGA: Asynchronous parallel SAGA. In *Artificial Intelligence and Statistics*, pages 46–54, 2017.

Yin Tat Lee and Aaron Sidford. Efficient accelerated coordinate descent methods and faster algorithms for solving linear systems. In *FOCS*, 2013.

David D Lewis, Yiming Yang, Tony G Rose, and Fan Li. RCV1: A new benchmark collection for text categorization research. *Journal of machine learning research*, 5(Apr):361–397, 2004.

Boyue Li, Shicong Cen, Yuxin Chen, and Yuejie Chi. Communication-efficient distributed optimization in networks with gradient tracking and variance reduction. In *International Conference on Artificial Intelligence and Statistics*, pages 1662–1672. PMLR, 2020a.

Huan Li and Zhouchen Lin. Revisiting extra for smooth distributed optimization. *SIAM Journal on Optimization*, 30(3):1795–1821, 2020.

Huan Li, Cong Fang, Wotao Yin, and Zhouchen Lin. A sharp convergence rate analysis for distributed accelerated gradient methods. *arXiv preprint arXiv:1810.01053*, 2018.

Huan Li, Cong Fang, Wotao Yin, and Zhouchen Lin. Decentralized accelerated gradient methods with increasing penalty parameters. *IEEE Transactions on Signal Processing*, 68:4855–4870, 2020b.

Huan Li, Zhouchen Lin, and Yongchun Fang. Optimal accelerated variance reduced extra and diging for strongly convex and smooth decentralized optimization. *arXiv preprint arXiv:2009.04373*, 2020c.

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

Zhi Li, Wei Shi, and Ming Yan. A decentralized proximal-gradient method with network independent step-sizes and separated convergence rates. *IEEE Transactions on Signal Processing*, 2019.

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.

Hongzhou Lin, Julien Mairal, and Zaid Harchaoui. A universal catalyst for first-order optimization. In *Advances in Neural Information Processing Systems*, pages 3384–3392, 2015a.

Hongzhou Lin, Julien Mairal, and Zaid Harchaoui. Catalyst acceleration for first-order convex optimization: from theory to practice. *Journal of Machine Learning Research*, 18 (1):7854–7907, 2017.

Qihang Lin and Lin Xiao. An adaptive accelerated proximal gradient method and its homotopy continuation for sparse optimization. *Computational Optimization and Applications*, 60(3):633–674, Apr 2015.

Qihang Lin, Zhaosong Lu, and Lin Xiao. An accelerated proximal coordinate gradient method. In *Advances in Neural Information Processing Systems*, pages 3059–3067, 2014.

Qihang Lin, Zhaosong Lu, and Lin Xiao. An accelerated randomized proximal coordinate gradient method and its application to regularized empirical risk minimization. *SIAM Journal on Optimization*, 25(4):2244–2273, 2015b.

Tao Lin, Sebastian U Stich, Kumar Kshitij Patel, and Martin Jaggi. Don't use large mini-batches, use local sgd. In *International Conference on Learning Representations*, 2019.

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.

Nicolas Loizou, Michael Rabbat, and Peter Richtárik. Provably accelerated randomized gossip algorithms. In *ICASSP 2019-2019 ieee international conference on acoustics, speech and signal processing (icassp)*, pages 7505–7509. IEEE, 2019.

Haihao Lu, Robert M Freund, and Yurii Nesterov. Relatively smooth convex optimization by first-order methods, and applications. *SIAM Journal on Optimization*, 28(1):333–354, 2018.

Wentian Lu and Gerome Miklau. Exponential random graph estimation under differential privacy. In *KDD*, 2014.

Zhaosong Lu and Lin Xiao. On the complexity analysis of randomized block-coordinate descent methods. *Mathematical Programming*, 152(1-2):615–642, 2015.

Chenxin Ma, Virginia Smith, Martin Jaggi, Michael I. Jordan, Peter Richtárik, and Martin Takáč. Adding vs. averaging in distributed primal-dual optimization. In *Proceedings of the International Conference on Machine Learning*, pages 1973–1982, 2015.

Chenxin Ma, Virginia Smith, Martin Jaggi, Michael I. Jordan, Peter Richtárik, and Martin Takáč. Distributed optimization with arbitrary local solvers. *Optimization Methods and Software*, 32(4):813–848, 2017.

Dhruv Mahajan, Nikunj Agrawal, S. Sathiya Keerthi, Sundararajan Sellamanickam, and Leon Bottou. An efficient distributed learning algorithm based on effective local functional approximations. *Journal of Machine Learning Research*, 19(74):1–37, 2018.

Julien Mairal. Optimization with first-order surrogate functions. In *International Conference on Machine Learning*, pages 783–791. PMLR, 2013.

Horia Mania, Xinghao Pan, Dimitris Papailiopoulos, Benjamin Recht, Kannan Ramchandran, and Michael I Jordan. Perturbed iterate analysis for asynchronous stochastic optimization. *SIAM Journal on Optimization*, 27(4):2202–2229, 2017.

Marie Maros and Joakim Jaldén. Panda: A dual linearly converging method for distributed optimization over time-varying undirected graphs. In *2018 IEEE Conference on Decision and Control (CDC)*, pages 6520–6525. IEEE, 2018.

H Brendan McMahan et al. Advances and open problems in federated learning. *Foundations and Trends® in Machine Learning*, 14(1), 2021.

Michael Mitzenmacher and Eli Upfal. *Probability and computing: Randomized algorithms and probabilistic analysis*. Cambridge University Press, 2005.

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. *Journal of Machine Learning Research*, 17(1):2165–2199, 2016.

Daniel K Molzahn, Florian Dörfler, Henrik Sandberg, Steven H Low, Sambuddha Chakrabarti, Ross Baldick, and Javad Lavaei. A survey of distributed optimization and control algorithms for electric power systems. *IEEE Transactions on Smart Grid*, 8(6): 2941–2962, 2017.

Giorgi Nadiradze, Amirmojtaba Sabour, Dan Alistarh, Aditya Sharma, Ilia Markov, and Vitaly Aksenov. Swarmsgd: Scalable decentralized sgd with local updates. *arXiv preprint arXiv:1910.12308*, 2019.

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. Distributed gradient methods for convex machine learning problems in networks: Distributed optimization. *IEEE Signal Processing Magazine*, 37(3):92–101, 2020.

Angelia Nedić and Alex Olshevsky. Stochastic gradient-push for strongly convex functions on time-varying directed graphs. *IEEE Transactions on Automatic Control*, 61(12):3936–3947, 2016.

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), 2017.

Arkadi Semenovič Nemirovski and David Borisovich Yudin. Problem complexity and method efficiency in optimization. 1983.

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

Yu Nesterov. Gradient methods for minimizing composite functions. *Mathematical Programming*, 140(1):125–161, 2013a.

Yurii Nesterov. *Introductory Lectures on Convex Optimization: A Basic Course*. Kluwer, Boston, 2004.

Yurii Nesterov. Gradient methods for minimizing composite functions. *Mathematical Programming, Ser. B*, 140:125–161, 2013b.

Yurii Nesterov. *Introductory Lectures on Convex Optimization: A Basic Bourse*, volume 87. Springer Science & Business Media, 2013c.

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.

Yurii E Nesterov. A method for solving the convex programming problem with convergence rate o (1/kˆ 2). In *Dokl. akad. nauk Sssr*, volume 269, pages 543–547, 1983.

Ivano Notarnicola and Giuseppe Notarstefano. Asynchronous distributed optimization via randomized dual proximal gradient. *IEEE Transactions on Automatic Control*, 62(5): 2095–2106, 2016.

Alex Olshevsky. Linear time average consensus and distributed optimization on fixed graphs. *SIAM Journal on Control and Optimization*, 55(6):3990–4014, 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.

Neal Parikh and Stephen Boyd. Proximal algorithms. *Foundations and Trends in Optimization*, 2014.

Pedro C Pinto, Patrick Thiran, and Martin Vetterli. Locating the source of diffusion in large-scale networks. *Physical Review Letters*, 2012.

Boris Pittel. On spreading a rumor. *SIAM Journal on Applied Mathematics*, 47(1):213–223, 1987.

Boris T Polyak. Some methods of speeding up the convergence of iteration methods. *Ussr computational mathematics and mathematical physics*, 4(5):1–17, 1964.

Boris T Polyak and Anatoli B Juditsky. Acceleration of stochastic approximation by averaging. *SIAM journal on control and optimization*, 30(4):838–855, 1992.

Shi Pu and Angelia Nedić. Distributed stochastic gradient tracking methods. *Mathematical Programming*, pages 1–49, 2020.

Shi Pu, Alex Olshevsky, and Ioannis Ch Paschalidis. Asymptotic network independence in distributed stochastic optimization for machine learning: Examining distributed and centralized stochastic gradientic descent. *IEEE Signal Processing Magazine*, 37(3):114–122, 2020.

Xun Qian, Zheng Qu, and Peter Richtárik. Saga with arbitrary sampling. In *International Conference on Machine Learning*, pages 5190–5199. PMLR, 2019a.

Xun Qian, Zheng Qu, and Peter Richtárik. L-svrg and l-katyusha with arbitrary sampling. *arXiv preprint arXiv:1906.01481*, 2019b.

Guannan Qu and Na Li. Accelerated distributed nesterov gradient descent. *IEEE Transactions on Automatic Control*, 65(6):2566–2581, 2019.

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*, 2011.

Sashank J Reddi, Ahmed Hefny, Suvrit Sra, Barnabás Pöczos, and Alex Smola. On variance reduction in stochastic gradient descent and its asynchronous variants. In *Proceedings of the 28th International Conference on Neural Information Processing Systems-Volume 2*, pages 2647–2655, 2015.

Sashank J. Reddi, Jakub Konečnỳ, Peter Richtárik, Barnabás Póczós, and Alex Smola. AIDE: Fast and communication efficient distributed optimization. *arXiv preprint arXiv:1608.06879*, 2016.

Dominic Richards and Patrick Rebeschini. Graph-dependent implicit regularisation for distributed stochastic subgradient descent. *Journal of Machine Learning Research*, 21(2020), 2020.

Peter Richtárik and Martin Takáč. Iteration complexity of randomized block-coordinate descent methods for minimizing a composite function. *Mathematical Programming*, 144 (1):1–38, 2014.

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

Herbert Robbins and Sutton Monro. A stochastic approximation method. *The annals of mathematical statistics*, pages 400–407, 1951.

R. T. Rockafellar. *Convex Analysis*. Princeton University Press, 1970.

Alexander Rogozin and Alexander Gasnikov. Penalty-based method for decentralized optimization over time-varying graphs. In *International Conference on Optimization and Applications*, pages 239–256. Springer, 2020.

Alexander Rogozin, César A Uribe, Alexander V Gasnikov, Nikolay Malkovsky, and Angelia Nedić. Optimal distributed convex optimization on slowly time-varying graphs. *IEEE Transactions on Control of Network Systems*, 7(2):829–841, 2019.

David Ruppert. Efficient estimations from a slowly convergent robbins-monro process. Technical report, Cornell University Operations Research and Industrial Engineering, 1988.

Ankan Saha and Ambuj Tewari. On the nonasymptotic convergence of cyclic coordinate descent methods. *SIAM Journal on Optimization*, 23(1):576–601, 2013.

Sujay Sanghavi, Bruce Hajek, and Laurent Massoulié. Gossiping with multiple messages. *IEEE Transactions on Information Theory*, 53(12):4640–4654, 2007.

Ali H Sayed. Adaptation, learning, and optimization over networks. *Foundations and Trends in Machine Learning*, 7(ARTICLE):311–801, 2014.

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

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, 2017b.

Kevin Scaman, Francis Bach, Sébastien Bubeck, Laurent Massoulié, and Yin Tat Lee. Optimal algorithms for non-smooth distributed optimization in networks. In *Advances in Neural Information Processing Systems*, pages 2745–2754, 2018.

Kevin Scaman, Francis Bach, Sébastien Bubeck, Yin Lee, and Laurent Massoulié. Optimal convergence rates for convex distributed optimization in networks. *Journal of Machine Learning Research*, 20:1–31, 2019.

Mark Schmidt, Nicolas Le Roux, and Francis Bach. Minimizing finite sums with the stochastic average gradient. *Mathematical Programming*, 162(1-2):83–112, 2017.

Othmane Sebbouh, Nidham Gazagnadou, Samy Jelassi, Francis Bach, and Robert Gower. Towards closing the gap between the theory and practice of svrg. In *Conference on Neural Information Processing Systems*, 2019.

Devavrat Shah. *Gossip algorithms*. Now Publishers Inc, 2009.

Devavrat Shah and Tauhid Zaman. Rumors in a network: Who's the culprit? *IEEE Transactions on Information Theory*, 2011.

Shai Shalev-Shwartz. Sdca without duality, regularization, and individual convexity. In *International Conference on Machine Learning*, pages 747–754. PMLR, 2016.

Shai Shalev-Shwartz and Tong Zhang. Stochastic dual coordinate ascent methods for regularized loss minimization. *Journal of Machine Learning Research*, 14(Feb):567–599, 2013.

Shai Shalev-Shwartz and Tong Zhang. Accelerated proximal stochastic dual coordinate ascent for regularized loss minimization. In *International Conference on Machine Learning*, 2014.

Ohad Shamir, Nati Srebro, and Tong Zhang. Communication-efficient distributed optimization using an approximate newton-type method. In *International conference on machine learning*, pages 1000–1008. PMLR, 2014.

Zebang Shen, Aryan Mokhtari, Tengfei Zhou, Peilin Zhao, and Hui Qian. Towards more efficient stochastic decentralized learning: Faster convergence and sparse communication. In *International Conference on Machine Learning*, pages 4631–4640, 2018.

Wei Shi, Qing Ling, Kun Yuan, Gang Wu, and Wotao Yin. On the linear convergence of the admm in decentralized consensus optimization. *IEEE Transactions on Signal Processing*, 62(7):1750–1761, 2014.

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, 2015a.

Wei Shi, Qing Ling, Gang Wu, and Wotao Yin. A proximal gradient algorithm for decentralized composite optimization. *IEEE Transactions on Signal Processing*, 63(22):6013–6023, 2015b.

Virginia Smith, Simone Forte, Ma Chenxin, Martin Takáč, Michael I Jordan, and Martin Jaggi. Cocoa: A general framework for communication-efficient distributed optimization. *Journal of Machine Learning Research*, 18:230, 2018.

Sebastian U Stich. Local sgd converges fast and communicates little. *International Conference on Learning Representations*, 2018.

Haipei Sun, Xiaokui Xiao, Issa Khalil, Yin Yang, Zhan Qin, Hui Wang, and Ting Yu. Analyzing subgraph statistics from extended local views with decentralized differential privacy. In *CCS*, 2019a.

Y Sun, A Daneshmand, and G Scutari. Distributed optimization based on gradient-tracking revisited: Enhancing convergence rate via surrogation. *arXiv preprint arXiv:1905.02637*, 2019b.

Ying Sun, Gesualdo Scutari, and Daniel Palomar. Distributed nonconvex multiagent optimization over time-varying networks. In *2016 50th Asilomar Conference on Signals, Systems and Computers*, pages 788–794. IEEE, 2016.

Hanlin Tang, Shaoduo Gan, Ce Zhang, Tong Zhang, and Ji Liu. Communication compression for decentralized training. *arXiv preprint arXiv:1803.06443*, 2018a.

Hanlin Tang, Xiangru Lian, Ming Yan, Ce Zhang, and Ji Liu. D^2: Decentralized training over decentralized data. In *International Conference on Machine Learning*, pages 4855–4863, 2018b.

Ye Tian, Ying Sun, and Gesualdo Scutari. Asy-sonata: Achieving linear convergence in distributed asynchronous multiagent optimization. In *2018 56th Annual Allerton Conference*

on Communication, Control, and Computing (Allerton), pages 543–551. IEEE, 2018.

Joel A. Tropp. An introduction to matrix concentration inequalities. *Foundations and Trends in Machine Learning*, 8(1-2):1–230, 2015.

César A Uribe, Soomin Lee, Alexander Gasnikov, and Angelia Nedić. A dual approach for optimal algorithms in distributed optimization over networks. *Optimization Methods and Software*, 2020.

Paul Vanhaesebrouck, Aurélien Bellet, and Marc Tommasi. Decentralized collaborative learning of personalized models over networks. In *AISTATS*, 2017.

Sharan Vaswani, Francis Bach, and Mark Schmidt. Fast and faster convergence of sgd for over-parameterized models and an accelerated perceptron. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pages 1195–1204. PMLR, 2019.

Roman Vershynin. *High-Dimensional Probability, An Introduction with Applications in Data Science*. Cambridge University Press, 2019.

Bin Wang, Hongyu Jiang, Jun Fang, and Huiping Duan. A proximal admm for decentralized composite optimization. *IEEE Signal Processing Letters*, 25(8):1121–1125, 2018a.

Jialei Wang and Lin Xiao. Exploiting strong convexity from data with primal-dual first-order algorithms. In *International Conference on Machine Learning*, pages 3694–3702, 2017.

Jialei Wang and Tong Zhang. Utilizing second order information in minibatch stochastic variance reduced proximal iterations. *Journal of Machine Learning Research*, 20(42):1–56, 2019.

Shusen Wang, Farbod Roosta-Khorasani, Peng Xu, and Michael W Mahoney. GIANT: Globally improved approximate Newton method for distributed optimization. In *Advances in Neural Information Processing Systems*, pages 2332–2342, 2018b.

Lin Xiao and Stephen Boyd. Optimal scaling of a gradient method for distributed resource allocation. *Journal of optimization theory and applications*, 129(3):469–488, 2006.

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.

Lin Xiao, Adams Wei Yu, Qihang Lin, and Weizhu Chen. DSCOVR: Randomized primal-dual block coordinate algorithms for asynchronous distributed optimization. *Journal of Machine Learning Research*, 20(43):1–58, 2019a.

Lin Xiao, Adams Wei Yu, Qihang Lin, and Weizhu Chen. DSCOVR: Randomized primal-dual block coordinate algorithms for asynchronous distributed optimization. *Journal of Machine Learning Research*, 20(43):1–58, 2019b.

Ran Xin, Soummya Kar, and Usman A Khan. Decentralized stochastic optimization and machine learning: A unified variance-reduction framework for robust performance and fast convergence. *IEEE Signal Processing Magazine*, 37(3):102–113, 2020a.

Ran Xin, Soummya Kar, and Usman A Khan. Gradient tracking and variance reduction for decentralized optimization and machine learning. *arXiv preprint arXiv:2002.05373*, 2020b.

Ran Xin, Usman A Khan, and Soummya Kar. Variance-reduced decentralized stochastic optimization with accelerated convergence. *IEEE Transactions on Signal Processing*, 68: 6255–6271, 2020c.

Jinming Xu, Shanying Zhu, Yeng Chai Soh, and Lihua Xie. Augmented distributed gradient methods for multi-agent optimization under uncoordinated constant stepsizes. In *2015*

54th IEEE Conference on Decision and Control (CDC), pages 2055–2060. IEEE, 2015.

Jinming Xu, Ye Tian, Ying Sun, and Gesualdo Scutari. Accelerated primal-dual algorithms for distributed smooth convex optimization over networks. In *International Conference on Artificial Intelligence and Statistics*, pages 2381–2391. PMLR, 2020a.

Jinming Xu, Ye Tian, Ying Sun, and Gesualdo Scutari. Distributed algorithms for composite optimization: Unified and tight convergence analysis. *arXiv preprint arXiv:2002.11534*, 2020b.

Jinming Xu, Ye Tian, Ying Sun, and Gesualdo Scutari. A unified algorithmic framework for distributed composite optimization. In *2020 59th IEEE Conference on Decision and Control (CDC)*, pages 2309–2316. IEEE, 2020c.

Haishan Ye, Luo Luo, Ziang Zhou, and Tong Zhang. Multi-consensus decentralized accelerated gradient descent. *arXiv preprint arXiv:2005.00797*, 2020.

Hsiang-Fu Yu, Hung-Yi Lo, Hsun-Ping Hsieh, Jing-Kai Lou, Todd G McKenzie, Jung-Wei Chou, Po-Han Chung, Chia-Hua Ho, Chun-Fu Chang, Yin-Hsuan Wei, et al. Feature engineering and classifier ensemble for KDD cup 2010. In *KDD Cup*, 2010.

Xiao-Tong Yuan and Ping Li. On convergence of distributed approximate Newton methods: Globalization, sharper bounds and beyond. *arXiv preprint arXiv:1908.02246*, 2019.

Xiao-Tong Yuan and Ping Li. On convergence of distributed approximate newton methods: Globalization, sharper bounds and beyond. *Journal of Machine Learning Research*, 21 (206):1–51, 2020.

Yun Zeng, Augustin Chaintreau, Don Towsley, and Cathy H Xia. Throughput scalability analysis of fork-join queueing networks. *Operations Research*, 66(6):1728–1743, 2018.

Jiaqi Zhang and Keyou You. Decentralized stochastic gradient tracking for empirical risk minimization. *arXiv preprint arXiv:1909.02712*, 2019.

Yuchen Zhang and Lin Xiao. DiSCO: Distributed optimization for self-concordant empirical loss. In *International Conference on Machine Learning*, pages 362–370, 2015.

Yuchen Zhang and Lin Xiao. Communication-efficient distributed optimization of self-concordant empirical loss. In *Large-Scale and Distributed Optimization*, number 2227 in Lecture Notes in Mathematics, chapter 11, pages 289–341. Springer, 2018.

Kaiwen Zhou. Direct acceleration of saga using sampled negative momentum. In *Artificial Intelligence and Statistics*, 2019.

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.