Review of Research Paper number 11

Raphael BERNAS & Maxime CORLAY

version: April 2, 2025

1 Introduction

The paper referred to through this report is "Optimal Convergence Rates for Convex Distributed Optimization in Networks" by Kevin Scaman, Francis Bach et al. [2]
The code for the practical parts can be found here : https://github.com/Raphael-Bernas/Distributed-Networks-Convex-Optimization

1.1 Summary of the paper

This paper inscribes itself in the peculiar settings of optimal convergence rates study. Here, we are not trying to assess whether we converge and at what cost. We are trying to find the best algorithm for each set of assumptions in order to derive the best convergence rate. Thus, the authors will obviously study an upper-bound on convergence rate but also a lower-bound (in order to prove that we are optimal with respect to a fixed constant). These works solve cases that were not developed in Nesterov's works and in the recent study "Decentralized and Parallelized Primal and Dual Accelerated Methods for Stochastic Convex Programming Problems" by Darina Dvinskikh and Aleksandr Gasnikov[1] [1]. Their goal is more to give a general description of optimal convergence rates (with respect to assumptions), than providing new algorithm to achieve this. Nonetheless, they still introduce and prove their own adapted optimal algorithms convergence rates.
Most of their findings are proposed in the form of error-dependant bound on time but can easily be reversed to get a bound on the error.
To dive a bit more in the paper let us detail quickly how it is divided :
First, the paper is divided between centralized and decentralized methods. For each of these, the authors study the convex case, then derive results for the strongly-convex case and finally detail the smooth case. Note that we always have Local Lipschitz continuity and convexity[2]. We give in this summary a first table of what they achieve in terms of times (In order for it to be easily readable, we have only given the dependance in term of the error ε) : See Table 1

1.2 Problem formulation

Let us suppose that we have n devices, each owning an objective function f_i which is private. They want to solve a general problem but without sharing their private objective function. There are two general ways to tackle such problems : in a centralized manner and in a decentralized one. Centralized means that there exists a node which is linked to all others (some kind of cloud). Such a node is meant to retrieve, process and share all outputs given by other nodes. Decentralized means that we are at a local level, where neighbors share with each other some information. This paper tries to solve the general distributed settings of :

$$\min_{\theta \in \mathbb{R}^d} \sum_{i=1}^{n} f_i(\theta) \tag{Centr}$$

[1] Indeed, for centralized algorithm, they prove that a naive Nesterov accelerated method leads to optimal convergences in smooth case. And they refer to Darina Dvinskikh and Aleksandr Gasnikov for decentralized smooth case. For all the other remaining cases, they adapt and improve algorithm in order to achieve optimal convergence rates.

[2] Convexity is the setting of this paper while local Lipshcitz continuity is required for non-smooth cases and is automatically true for smooth cases on a bounded set.

Setting	Lower bound (in terms of ε)	Upper bound (in terms of ε)
Lipschitz continuity + convexity		
Centr.	$\Omega\big(\varepsilon^{-2} + \varepsilon^{-1}\big)$	$O\big(\varepsilon^{-2} + \varepsilon^{-1}\big)$
Decentr.	$\Omega\big(\varepsilon^{-2} + \varepsilon^{-1}\big)$	$O\big(\varepsilon^{-2} + \varepsilon^{-1}\big)$
Lipschitz continuity + strong convexity		
Centr.	$\Omega\big(\varepsilon^{-1} + \varepsilon^{-\frac{1}{2}}\big)$	$O\big(\varepsilon^{-1} + \varepsilon^{-\frac{1}{2}} log(\frac{1}{\varepsilon})\big)$
Decentr.	$\Omega(\varepsilon^{-1} + \varepsilon^{-\frac{1}{2}})$	$O(\varepsilon^{-1} + \varepsilon^{-\frac{1}{2}})$
Smoothness + convexity		
Centr.	$\Omega(\varepsilon^{-\frac{1}{2}})$	$O(\varepsilon^{-\frac{1}{2}})$
Decentr.	$\Omega(\varepsilon^{-\frac{1}{2}})$	$O(\varepsilon^{-\frac{1}{2}})$
Smoothness + strong convexity		
Centr.	$\Omega(log(\frac{1}{\varepsilon}))$	$O(log(\frac{1}{\varepsilon}))$
Decentr.	$\Omega(log(\frac{1}{\varepsilon}))$	$O(log(\frac{1}{\varepsilon}))$

Table 1: Illustrative table retaining only the ε-dependence of each convergence rate. Recall that when we are proceedings in term of ε, we have that $log(\frac{1}{\varepsilon}) >> \varepsilon^{-\frac{1}{2}} >> \varepsilon^{-1} >> \varepsilon^{-2}$.

In the centralized setting we can keep this formulation in order to proceed with the problem. However, in the decentralized setting, we try to address :

$$\begin{aligned} \min_{\Theta \in \mathbb{R}^{n \times d}} \quad & \sum_{i=1}^{n} f_i(\Theta_i) \\ \text{subject to} \quad & \Theta_i = \Theta_j, \quad \forall\, i, j \in \{1, \dots, n\}. \end{aligned} \tag{Decentr}$$

Do note that this is a very general setting which allows for any kind of cooperative problem to be modelized.

1.3 Assumptions

A first and general assumption that is set for all the paper is that we search $\theta \in \mathcal{K} \subset \mathbb{R}^d$ where $\mathcal{K}$ is a bounded convex set.
Then, the assumptions are divided betweens two types : regularity assumptions, which impact the objective functions class, and graph assumptions, which impact the interaction between devices.

1.3.1 Regularity assumptions

First of all, throughout the whole paper, we are in a convex case. Furthermore, there are four types of settings (which are always studied in the same order):

1. Lipschitz continuity & convexity
2. Lipschitz continuity & strong-convexity
3. Smoothness & convexity
4. Smoothness & strong-convexity

Each of those assumptions apply to all f_i functions. This is what they call "local regularity". Global regularity would be to expect some regularity from the sum of those f_i. But to cite them "Finding an optimal distributed algorithm for global regularity is, to our understanding, a much more challenging task and is left for future work.".

1.3.2 Graph assumptions

For the centralized case, there is not much assumptions other than the fact that their exist a octopus node which is linked with all the others. Then they just proceed to compute a spanning tree on the graph

so that every nodes share information with the root node.
But for the decentralized case, we need some assumptions on our graph and its weight matrix. Let us note $\mathcal{G} = (\mathcal{V}, \mathcal{E}, W)$ our communication graph. We will need the following hypothesis on $\mathcal{G}$:

- W is symmetric and positive semi-definite.
- $Ker(W) = Span((1, 1, \cdots, 1)^T)$, the only vectors in the kernel of W are constant vectors.
- For all $(i, j) \notin \mathcal{E}$, $W_{ij} = 0$.

In some theorem we have another hypothesis which is that the mixed time (defined as the inversed square root of the normalized eigengap) is bigger than one, but this is always true thanks to those above assumptions.

1.3.3 Some more assumptions

Moreover, for the lower bounding theorem, they require to assume that they apply black-box procedure (which is the case for all the algorithms we have seen so far in SOD314 and the algorithm they test here). This basically means that they need some more assumptions on the devices, there oracles and the communication between them. These are classical assumptions (present in the distributed setting, just like in SOD314 courses) and we won't detail all of them but we note that each devices can access its gradient and can communicate information to its neighbors in a fixed time τ, etc.

2 Result: theory and practice

2.1 Theory

The paper has many intermediary results and even defines completely the algorithm used to derive theoretical convergence rates. But they don't focus on the algorithm. Indeed, they precisely search to find the optimal convergence rates. To do so, they follow this general process : first, they set themselves in one of the eight possible settings (centralized/decentralized $\times$ one of the four regularity assumptions). Then, they derive a lower-bound for any kind of black-box algorithm under those assumptions. Then, they choose an optimal algorithm which they prove (or refer to someone else's proof) to attain some upper-bound on time for ε-error convergence and conclude on a quite tight optimal convergence rate. Here, we will provide those main theorems in a simpler way (without detailling the algorithm, to learn more about those algorithms, please refer to the Practice section). Let us first introduce some notations (those are different from the original paper and are based on SOD314 course) :

- We note C_g and C_i, the Lipschitz constant of the general and local function ($F = \sum_{i=1}^n f_i$ and f_i)
- We note L_g and L_i, the smooth constant of the general and local function.
- We note α_g and α_i, the strong-convexity constant of the general and local function.
- We note R, the ray of the smallest ball that contain $\mathcal{K}$ and the initialization point θ_0.
- We note Δ the diameter of our communication graph (even in the centralized case). Note that Δ is linked to the eigengap (bounded by it). This links their results with the one achieved in SOD314 lectures.
- $\tilde{\Delta} = \sqrt{\frac{\lambda_1(W)}{\lambda_{n-1}(W)}}$ where λ are sorted eigen values of W (the biggest is the first).
- We note τ the irreducible time needed for connected devices to exchange their informations.
- Finally, we define $C_l = \sqrt{\frac{1}{n}\sum_i C_i^2}$, $\alpha_l = \min_i \alpha_i$. - $L_l = \max_i L_i$.

We will now detail the main theorem derived in the paper. Please note that for each theorem, the assumptions follow the assumption scheme detailed in section 1.3.

2.1.1 Centralized

The optimal algorithms used in this section are an adapted form of Distributed Randomized Algorithm (DRS) for non-smooth problems and a Naive form of Accelerated Gradient Descent (AGD) for smooth problems.

Lipschitz continuity & convexity :

- *Lower bound* : (Theorem 8) Under concerned assumptions, we have for black-box algorithm, a lower bound : $\Omega\left(\left(\frac{RC_g}{\varepsilon}\right)^2 + \frac{RC_g}{\varepsilon}\Delta\tau\right)$.

- *Upper bound* : (Theorem 2) Under concerned assumptions, we have for a carefully crafted optimal algorithm, an upper bound : $O\left(\left(\frac{RC_g}{\varepsilon}\right)^2 + \frac{RC_g}{\varepsilon}(1+\Delta\tau)d^{1/4}\right)$.

Lipschitz continuity & strong-convexity :

- *Lower bound* : (Theorem 9) Under concerned assumptions, we have for black-box algorithm, a lower bound : $\Omega\left(\frac{C_g^2}{\alpha_g\varepsilon} + \sqrt{\frac{C_g^2}{\alpha_g\varepsilon}}\Delta\tau\right)$.

- *Upper bound* : (Theorem 6) Under concerned assumptions, we have for a carefully crafted optimal algorithm, an upper bound : $O\left(\frac{C_g^2}{\alpha_g\varepsilon} + \sqrt{\frac{C_g^2}{\alpha_g\varepsilon}}(1+\Delta\tau)d^{1/4}\log\left(\frac{1}{\varepsilon}\right)\right)$.

Smoothness & convexity :

- *Tight bound* : (Theorem 11) Under concerned assumptions, we have for an adapted AGD, the tight bound : $\Theta\left(R\sqrt{\frac{L_g}{\varepsilon}}(1+\Delta\tau)\right)$.

Smoothness & strong-convexity :

- *Tight bound* : (Theorem 12) Under concerned assumptions, we have for an adapted AGD, the tight bound : $\Theta\left(\sqrt{\frac{L_g}{\alpha_g}}\ln\left(\frac{1}{\varepsilon}\right)(1+\Delta\tau)\right)$.

2.1.2 Decentralized

The optimal algorithms used in this section are an adapted form of Multi-Step Primal Dual (MSPD) for non-smooth & convex problem, an adapted form of Restart Sliding [1] (R-S) for strongly-convex and an adapted Penalty Similar Triangles Methods [1] (PSTM) for smooth problem.

Lipschitz continuity & convexity :

- *Tight bound* : (Theorem 14) Under concerned assumptions, we have for an adapted MSPD, a tight bound : $\Theta\left(\left(\frac{RC_l}{\varepsilon}\right)^2 + \frac{RC_l}{\varepsilon}\tilde{\Delta}\tau\right)$.

Lipschitz continuity & strong-convexity :

- *Tight bound* : (Theorem 15 - Results from [1]) Under concerned assumptions, we have for an adapted R-S, a tight bound : $\Theta\left(\frac{C_l^2}{\alpha_l\varepsilon} + \sqrt{\frac{C_l^2}{\alpha_l\varepsilon}}\tilde{\Delta}\tau\right)$.

Smoothness & convexity :

- *Tight bound* : (Theorem 16 - Results from [1]) Under concerned assumptions, we have for an adapted PSTM, the tight bound : $\Theta\left(R\sqrt{\frac{L_l}{\varepsilon}}(1+\tilde{\Delta}\tau)\right)$.

Smoothness & strong-convexity :

- *Tight bound* : (Theorem 17 - Results from [1]) Under concerned assumptions, we have for an adapted PSTM, the tight bound : $\Theta\left(\sqrt{\frac{L_l}{\alpha_l}}\ln\left(\frac{1}{\varepsilon}\right)(1+\tilde{\Delta}\tau)\right)$.

2.2 Practice

The code for those experiments can be found here : https://github.com/Raphael-Bernas/Distributed-Networks-Convex-Optimization

In this section, our goal is to realize a study on the algorithm proposed in the paper [2]. We have implemented the three algorithms which where introduced by the authors of the paper : Distributed Randomized Smoothing Algorithm for both convex and strongly convex cases. And Multi-Step Primal Dual Algorithm.

Note that the smooth case algorithm were not studied by the authors of this paper but by the ones of [1]. Thus, we need to set ourselves in a non-smooth setting. As the Kernel problem from SOD314 is clearly

smooth, we will need to adapt it a little bit :

In practical setting for the course SOD314 we tried to solve (note that here $d = m$) :

$$\min_{\alpha \in \mathbb{R}^m} f(\alpha) := \sum_{i=1}^{n} \frac{\sigma^2}{2} \alpha^\top K_{mm}^{(i)} \alpha + \frac{1}{2} \|y - K_{mm}^{(i)} \alpha\|_2^2 + \frac{\nu}{2} \|\alpha\|_2^2$$

Which is a smooth problem (even strongly convex with carefully crafted matrix and parameters). Here, we are going to change a bit the problem so that it is no longer smooth but so that we could still expect to obtain the same optimum (we changed the L2-norm for L1-norm in the loss function which compare labels with predictions):

$$\min_{\alpha \in \mathbb{R}^m} f_{NS}(\alpha) := \sum_{i=1}^{n} \frac{\sigma^2}{2} \alpha^\top K_{mm}^{(i)} \alpha + \|y - K_{mm}^{(i)} \alpha\|_1 + \frac{\nu}{2} \|\alpha\|_2^2$$

This problem is indeed non-smooth. That is because around 0, the L1-norm as a discontinuous gradient which results in an infinite Lipschitz constant of the gradient.

Then, we set ourselves in a ball of radius R : $\mathcal{K} = \mathcal{B}(0_m, R)$. And finally we derive useful constant (as explained before, we won't need smooth constant because the authors only solved non-smooth cases while smooth cases were studied in [1] even if they include their theoretical results in this paper).

$$C_g = \sum_{i=1}^{n} \left(\sigma^2 \lambda_{\max}(K_{mm}^{(i)}) R + \sqrt{m} \lambda_{\max}(K_{mm}^{(i)}) \right) + n\nu R.$$

$$L_g = \sigma^2 \sum_{i=1}^{n} \lambda_{\min}(K_{mm}^{(i)}) + n\,\nu$$

$$C_i = \sigma^2 \lambda_{\max}(K_{mm}^{(i)}) R + \sqrt{m} \lambda_{\max}(K_{mm}^{(i)}) + \nu R.$$

In fact, we won't need any other parameters for the case that the algorithms introduced in the paper solve.

For all three algorithms, we added hyperparameters which allow to restrict the number of steps (because in their paper, it is based on a given ε). Note that if we were to compute the algorithm without limited iterations, for our problem, it would take many days to achieve an error of 10^{-3}.

2.2.1 Centralized

For the non-smooth centralized case, we are given two optimal algorithms : Distributed Randomized Smoothing in convex and strongly-convex setting. The main and elegant idea behind those algorithms is that we can recover smoothness through resolving an intermediary problem :

$$f_{NS}^{\gamma}(\theta) = \mathbb{E}\big[f_{NS}(\theta + \gamma X)\big]$$

And using some lemma we get a bound on this function using the original problem f_{NS}.

In any case, we implemented the detailed methods (even if both have the same name and derive from the same method, they are implemented very differently, which will lead to highly different observations). Using the theoretical formula, we can estimate the error value ε based on the number of iterations we realize. This is named estimated epsilon in the following figure.

- *Convex :*

Refer to Figure 1. We achieved the following results in $40,000$ iterations which amounts for 43 min 28 s (on CPU AMD Ryzen 7 5800H Laptop). It is truly impressive how tight their bound is. We almost reached the theoretically expected $\varepsilon = 0.24$ (we get ≈ 0.21). This shows the power of theory, we know how to achieve a given ε even if it could take quite some time.

- *Strongly convex :*

For this algorithm, we took 76 min 13 s to obtain an error of 0.355 which is worse than our DRS convex case algorithm[3]. Furthermore, with a $\varepsilon = 0.0024$ the bound on estimated epsilon is bigger (this could be due to the fact that it is harder to derive a robust approximation of it practically speaking). A first deduction we can make is that the strongly convex algorithm is theoretically quicker than the convex

[3]This was obtained with $T = 2$ iterations and almost $50,000$ sub-iterations (iterations inside the iterations, done on intermediary variables). It is the reason why we did not add the graph here, with two values it was pretty much useless.

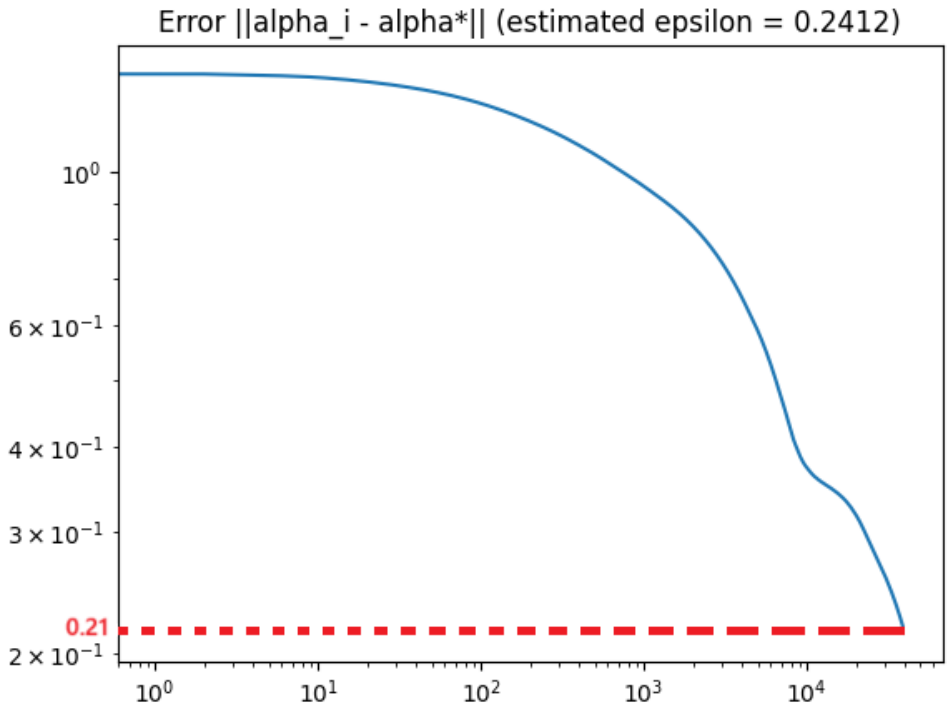

Figure 1: $\|\alpha_k^i - \alpha^*\|$ for DRS in convex case (estimated epsilon is the theoretical estimation given by the bound on the error in term of iteration time).

case algorithm but in practical setting, it becomes slower. And that is because there is a huge number of intermediary steps that we need to process for each iteration of the algorithm. This makes it less practical, and the worst part is that the number of intermediary steps is exponentially increasing while the stepsize becomes very small. This results in an underperforming process. The second observation is that this method is quite unstable. Indeed, if we do not perform enough intermediary steps (in the function we defined, you can limit the number of intermediary steps), then the whole process explodes. This is mainly due to Nesterov acceleration which allows far better theoretical convergence but results in unstable algorithm (because the momentum μ_t tends very slowly to 1). We advise you to refer to the notebook (our code) in order to find a more detailed explanation on that.

2.2.2 Decentralized

Refer to Figure 2. The algorithm has taken 24 min 39 s to finish. Here we observe an interesting phenomenum. A bouncing curve of error. It is probably due to the fact that we proceeded with a restricted number of sub-iterations (and not all the required sub-iterations). This possibly resulted in an introduced error - which is due to the way the optimum is computed[4]- propagated through the whole computation. But, if we only retain the best model computed so far, we see that we achieved an error of 0.039 (at time $k = 250$) which is near the value we expected from the theory $\varepsilon = 0.048$. This is once again a proof of how tight their bounds are (recall that they have given similar magnitude lower and upper bounds for this case).

However, it is possible that the "rebound" observed is due to the problem itself. Indeed, it is not clear whether the problem we introduced in order to solve this data fitting is good. And it could be highly dependant on the of points given, etc.

3 Conclusion

During the course SOD314, we discovered algorithm for which we had convergence rates thanks to multiple assumptions. One of those is smoothness, which was almost always present. Thus, the first important point for us is that this paper solves the case of non-smooth functions and with an elegant method which allows them to recover smoothness. Furthermore, this was not a focus of SOD314 but here, they derive

[4]For each iteration, the optimum is computed as a mean over all past iterations and over all agents.

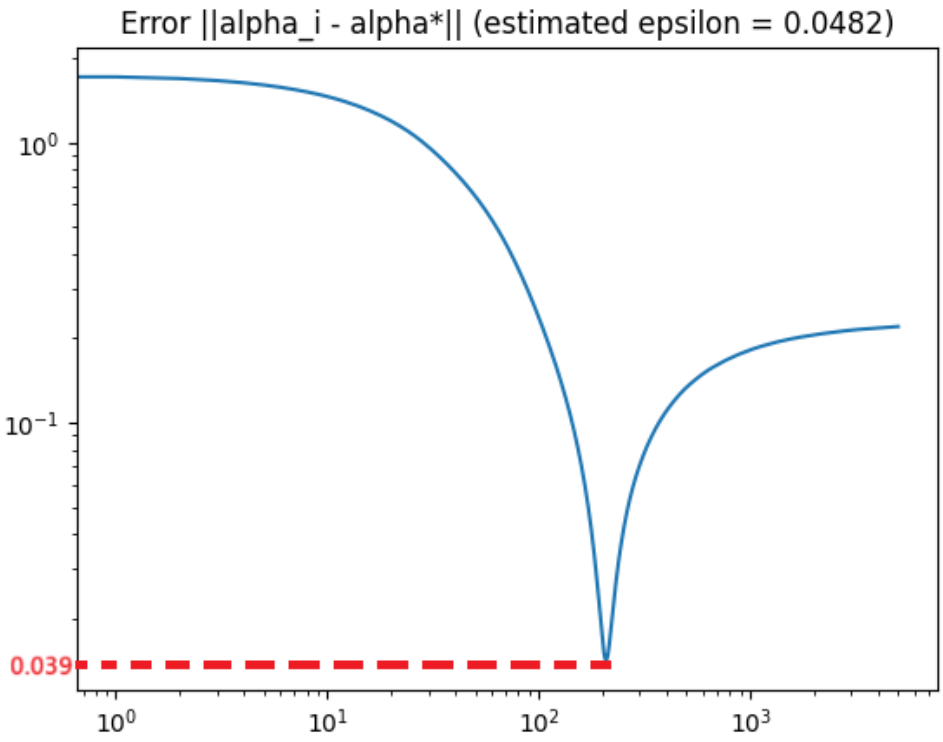

Figure 2: $\|\alpha_k^i - \alpha^*\|$ for MSPD in convex case (estimated epsilon is the theoretical estimation given by the bound on the error in term of iteration time).

optimal convergence rates for each case (if you suppose that you are using a black-box process). More surprisingly, their bound are tight (this means that there are not huge hidden constant in $O(\cdot)$, $\Omega(\cdot)$ and $\Theta(\cdot)$). We can now add to our collection new algorithms. But, unfortunately, those algorithms are much harder to implement than the ones from the course (which makes them less practical for a non-specialized person). And more importantly, they are less robust to hyperparameters variations. Indeed, with SAGA and other algorithms like this from the course, we could retrieve good results easily and even in cases where theory couldn't explain why. This is less the case for those. Thus, we do comprehend why they did not try an implementation (and some experimental tests) in their papers. Finally, the methods that were purely motivated by theory have still often proven to be useful for experiments, even in settings that the theory did not cover. This is why we believe that we should keep some ideas used here in the back of our mind !

References

[1] Darina Dvinskikh and Alexander Gasnikov. Decentralized and parallel primal and dual accelerated methods for stochastic convex programming problems, 2021.

[2] Kevin Scaman, Francis Bach, Sébastien Bubeck, Yin Tat Lee, and Laurent Massoulié. Optimal convergence rates for convex distributed optimization in networks. *Journal of Machine Learning Research*, 20(159):1–31, 2019.