Sparse Asynchronous Distributed Learning

Dmitry Grischenko[†], Franck Iutzeler[†], Massih-Reza Amini[‡]

Université Grenoble Alpes, CNRS, France
[†]Department of Statistics (LJK) [‡]Department of Computer Science (LIG)
`firstname.lastname@univ-grenoble-alpes.fr`

Abstract. In this paper, we propose an asynchronous distributed learning algorithm where parameter updates are performed by worker machines simultaneously on a local sub-part of the training data. These workers send their updates to a master machine that coordinates all received parameters in order to minimize a global empirical loss. The communication exchanges between workers and the master machine are generally the bottleneck of most asynchronous scenarios. We propose to reduce this communication cost by a sparsification mechanism which, for each worker machine, consists in randomly and independently choosing some local update entries that will not be transmitted to the master. We provably show that if the probability of choosing such local entries is high and that the global loss is strongly convex, then the whole process is guaranteed to converge to the minimum of the loss. In the case where this probability is low, we empirically show on three datasets that our approach converges to the minimum of the loss in most of the cases with a better convergence rate and much less parameter exchanges between the master and the worker machines than without using our sparsification technique.

Keywords: Asynchronous Optimization · Sparsification

1 Introduction

Given the tremendous production of data and the ever growing size of collections, there is a surge of interest in both Machine Learning and Optimization communities for the development of efficient and scalable distributed learning strategies. In such context, training observations are generally split over different computing nodes and learning is performed simultaneously where each node, also referred to as a *worker*, has its own memory and processing unit. Note that this is different from shared-memory parallel computing, where each worker machine can potentially have access to all available memory [17,9]. However, the bottleneck of distributed learning is the network bandwidth as for parameter tuning, information is exchanged across the nodes that are organized across a LAN. Most of the distributed algorithms perform parameter updates in a synchronized manner [2,4]. For these approaches, the slower worker machines may slow down the whole learning process as the faster ones have to wait all updates in order to terminate their computation and exchange information. Recently, many studies have focused on asynchronous distributed frameworks, where worker machines update their parameters simultaneously on a local sub-part of data and send their updated parameters to a master machine. The master integrates then all received parameters and broadcasts them back to each computing node for a new local update of their parameters [8,14]. The communication cost between workers and the master machine is generally prohibitive in asynchronous scenarios and most attention has been paid on reducing this cost [12,11,18]. In this way, asynchronous coordinate descent methods were proposed in [7,16]. These techniques are able to handle unbounded delays but they are based on decreasing stepsizes. To overcome this restriction, the recent works of [14,13] provide a delay-independent analysis technique that allows to integrate assumptions on the computing system with a constant stepsize and which was shown to outperform other asynchronous distributed strategies.

In this paper, we propose a first theoretically founded Sparse asynchrOnous Distributed leArning framework (called `SODA`). Our strategy aims to reduce the size of communications where just a part of the information is transmitted from a worker to the master in order to accelerate the convergence of delay-independent asynchronous distributed methods using a sparsification mechanism. This is done trough a sparsification mechanism that consists in choosing some local update entries that will not be transmitted to the master. Moreover, in the case of ℓ_1-regularized problems, we show that at the master level, update iterates identify some sparsity pattern in finite time with probability one, resulting in sparse downward communications from the master to the workers. Thus, communications in both directions (from the workers to the master and vice versa) become sparse. As a consequence, we leverage on this identification to improve our sparsification technique by preferably sampling the entries in the support of the master model. We show that in the case of strongly convex objectives, the convergence to the global empirical risk estimated over the whole training set is guaranteed when the probability of choosing such local entries is high. In the case where this probability is low, we empirically show on three datasets that our approach converges to the minimum of the loss in most of the cases with a better convergence rate than without using our sparsification strategy.

In the following section, we present our asynchronous distributed framework with sparse communication and prove in Section 3, that the approach is guaranteed to converge to the minimum of a strongly convex global empirical loss if entries of parameters transmitted to the master machine are randomly chosen with high probability. Section 4 describes experimental results that support this approach and Section 5 concludes this work by giving some pointers for some future work.

2 Asynchronous distributed learning with sparsification

In this section we present our asynchronous distributed algorithm with sparse communications. Section 2.1 introduces the notation and problem formulation for asynchrony, and Section 2.2 presents the algorithm.

2.1 Notations and Framework

We consider the following distributed setup where there are M worker machines, $i \in \{1, \ldots, M\}$, each of which contains a subset $\mathcal{S}_i \subset \mathcal{S}$ of the training set (i.e. $\mathcal{S} = \sqcup_{i=1}^{M} \mathcal{S}_i$). Learning over such scattered data leads to optimization problems with composite objective of the form:

$$\min_{\boldsymbol{w} \in \mathbb{R}^n} \mathcal{L}(\boldsymbol{w}) = \sum_{i=1}^{M} \pi_i f_i(\boldsymbol{w}) + \lambda_1 \|\boldsymbol{w}\|_1, \tag{1}$$

where $\boldsymbol{w} \in \mathbb{R}^n$ are the parameters shared over all computing machines, $m = |\mathcal{S}|$ is the size of the training set, $\pi_i = |\mathcal{S}_i|/m$ is the proportion of observations locally stored in worker machine i, $f_i(\boldsymbol{w}) = \frac{1}{|\mathcal{S}_i|} \sum_{j \in \mathcal{S}_i} \ell_j(\boldsymbol{w})$ is the local empirical risk estimated on the subset $\mathcal{S}_i$ for machine i; ℓ_j is a smooth loss function for the training example $j \in \mathcal{S}_i$.

In this setting, our algorithm carries out computations without waiting for slower machines to finish their jobs. Each worker machine performs computations based on outdated versions of the main variable that it has. The master machine gathers the workers inputs, updates the parameters at each communication and sends back the updated versions to the current worker with which it is in interaction. We formalize this framework as :

- *For the master.* We define k, as the number of updates that the master has received from any of the worker machines. Thus, at time k, the master receives some input from a worker, denoted by i^k, updates its global variables, $\overline{\boldsymbol{w}}^k$ and $\boldsymbol{w}^k$; and sends $\boldsymbol{w}^k$ back to worker i^k.
- *For a worker* $i \in \{1, \ldots, M\}$. At time k, let d_i^k be the elapsed moment since the last time the master has communicated with worker i ($d_i^k = 0$ iff the master gets updates from worker i at exactly time k, i.e. $i^k = i$). We also consider D_i^k as the elapsed time since the second last update. This means that, at time k, the last two moments that the worker i has communicated with the master are $k - d_i^k$ and $k - D_i^k$.

2.2 Sparsity mask selection

Each worker independently computes a gradient step on its local loss for a randomly drawn subset of coordinates only. More specifically, the master machine keeps track of the weighted average of the most recent workers outputs, computes the proximity operator of the regularizer at this average point, and sends this result back to the updating worker i^k.

At iteration k, the randomly drawn subset of entries that worker i^k updates at time k is called *mask* and is denoted by $\mathbf{M}_p^k$ (in uppercase bold, to emphasize that it is the *only* random variable in the algorithm). We propose to select all the coordinates that are nonzero in the last parameter $\boldsymbol{w}$ received from the master machine, called $\text{supp}(\boldsymbol{w})$, and select some random zero entries of $\boldsymbol{w}$ with a fixed probability p (Algorithm 1). Without sparsification (i.e. $p = 1$

Algorithm 1 SODA

```
Worker  i
Initialize $\boldsymbol{w}_i = \boldsymbol{w}_i^+ = \overline{\boldsymbol{w}}^0$
while not interrupted do
    Receive $\boldsymbol{w}$ from master
    Draw sparsity mask $\mathbf{M}_p$ as
    $\mathbb{P}[j \in \mathbf{M}_p] = \begin{cases} 1, & \text{if } j \in \mathrm{supp}(\boldsymbol{w}) \\ p, & \text{if } j \notin \mathrm{supp}(\boldsymbol{w}) \end{cases}$
    $\boldsymbol{w}_i^+ = [\boldsymbol{w} - \gamma \nabla f_i(\boldsymbol{w})]_{\mathbf{M}_p} + [\boldsymbol{w}_i]_{\overline{\mathbf{M}_p}}$
    $\Delta = \boldsymbol{w}_i^+ - \boldsymbol{w}_i$
    $\boldsymbol{w}_i = \boldsymbol{w}_i^+$
    Send $\Delta$ to master
end while
```

```
Master
Initialize $\overline{\boldsymbol{w}}^0$
while not converged do
    Receive $\Delta^k$ from agent $i^k$
    $\overline{\boldsymbol{w}}^k = \overline{\boldsymbol{w}}^{k-1} + \pi_{i^k} \Delta^k$
    $\boldsymbol{w}^k = \mathbf{prox}_{\gamma r}(\overline{\boldsymbol{w}}^k)$
    Send $\boldsymbol{w}^k$ to agent $i^k$
end while
Output $\boldsymbol{w}^k$
```

and $\mathbf{M}_1^k = \{1 \ldots, n\}$ at any time k), this iteration corresponds to the delay-independent asynchronous proximal-gradient algorithm proposed in [14].

The proposed algorithm SODA uses the following notation: for a vector of $\boldsymbol{w} \in \mathbb{R}^n$ and a subset S of $\{1, .., n\}$, $[\boldsymbol{w}]_S$ denotes the sparse vector where S is the set of non-null entries, for which they match those of $\boldsymbol{w}$, i.e. $([\boldsymbol{w}]_S)_{[i]} = \boldsymbol{w}_{[i]}$ if $i \in S$ and 0 otherwise. In addition, we denote by $\overline{\mathbf{M}_p} = \{i \in \{1, \ldots, n\} : i \notin \mathbf{M}_p\}$.

In algorithm 1, communications per iteration are (i) a blocking send/receive from a slave to the master (in blue) of size $|\mathbf{M}_p|$, and (ii) a blocking send/receive from the master to the last updating slave (in red) of the current iterate.

3 Theoretical analyses

In this section, we first bound the expected deviation of the distance between a current solution found at iteration k and the true minimizer of the global loss (1) and then present our main result.

3.1 General convergence result

Lemma 1 (Expected deviation [6]). *Suppose all functions $\{f_i\}_{i=1,..,M}$ are μ-strongly convex and L-smooth (i.e. differentiable with Lipschitz continuous gradient). Let $\mathbf{w}^\star$ be the minimizer of the loss (1), and take $\gamma \in (0, 2/(\mu + L)]$, then for all $k \in [k_s, k_{s+1})$:*

$$\mathbb{E}\|\boldsymbol{w}^k - \boldsymbol{w}^\star\|^2 \leq (1-\beta)^s \max_{i=1,..,M} \|\boldsymbol{w}_i^0 - \boldsymbol{w}_i^\star\|^2, \tag{2}$$

where $\boldsymbol{w}_i^\star = \boldsymbol{w}^\star - \gamma \nabla f_i(\boldsymbol{w}^\star); i \in \{1, \ldots, M\}$, and, the sequence of stopping times (k_s) defined iteratively as $k_0 = 0$ and

$$k_{s+1} = \min\left\{k : k - D_i^k \geq k_s \text{ for all } i\right\}, \tag{3}$$

and

$$\beta = 2\frac{\gamma\mu L}{\mu + L} - 1 + p.$$

This general result deserves several comments:

- The sequence (k_s) of stopping times is defined such that there are at least two updates of each worker between k_{s+1} and k_s. This sequence directly embeds the number of machines and the delays, and thus automatically adapts the various situations.
- Furthermore, we retrieve the convergence results of [13] in the case where there are no sparsification (i.e. $p = 1$), and if there is no delay, we recover the rate of vanilla proximal-gradient algorithm.
- Assuming that all machines are responsive (i.e. $s \to \infty$ when $k \to \infty$), the inequality (5) gives convergence if $\beta > 0$, i.e. $p > 1 - \alpha$. In other words, when we sample entries non-uniformly, we still have convergence if the probability of selection is big enough.
- Finally, when the problem is well-conditioned (i.e. $\mu \simeq L$ and thus $\alpha \simeq 1$), the algorithm is guaranteed to converge for any reasonable choice of p.

3.2 Sparsity Structure Identification and Effect on Communications

Recently, many studies have been conducted on introducing sparsity in the structure of parameters minimizing a learning objective with a ℓ_1-norm regularization term [1,3] and the identification of such sparsity structures with proximal gradient methods [5,15]. Unfortunately, proximal gradient algorithms with ℓ_1-norm regularization term featuring random values, as the mask that is used for sparsification of update, might not identify structure with probability one. However, as soon as an almost sure convergence is established, sparsity structure identification holds [10]. Taking into account that there is no almost sure convergence in our case let us adapt the identification result of [5] using the following Lemma.

Lemma 2 (Identification). *Suppose that that for any $\varepsilon > 0$ there exists iterate number K such that for any $k > K$ the average point $\|\overline{\boldsymbol{w}}^k - \overline{\boldsymbol{w}}^\star\|_2^2 < \varepsilon$ is ε-close to the final solution. Furthermore, let us assume that problem (1) is non-degenerate. That is :*

$$\left(\sum_{i=1}^{M} \pi_i \nabla f_i(\boldsymbol{w}^\star)\right)_{[j]} < \lambda_1 \qquad \text{for all } j \in \operatorname{supp}(\boldsymbol{w}^\star). \tag{4}$$

Then for any $k > K$, we have : $\operatorname{supp}(\boldsymbol{w}^k) = \operatorname{supp}(\boldsymbol{w}^\star)$.

Now we are ready to present the following theoretical result that explains the practical interest of using the `SODA` algorithm in the case where the identification property takes place.

Theorem 1. *Suppose that all functions* $\{f_i\}_{i=1,..,M}$ *in (Eq. 1) are* μ*-strongly convex and* L*-smooth. Let* $\gamma \in (0, 2/(\mu + L)]$*, then in the case of identification of Lemma 2,* $\operatorname{supp}(\boldsymbol{w}^k) = \operatorname{supp}(\boldsymbol{w}^\star)$*, we have the following inequality :*

$$\|\boldsymbol{w}^k - \boldsymbol{w}^\star\|^2 \leq (1-\alpha)^s C_i, \tag{5}$$

where $\alpha = 2\frac{\gamma\mu L}{\mu+L}$*, e*$C_i = (1-\alpha)^{-s_i}\|\boldsymbol{w}^{k_{s_i+1}} - \boldsymbol{w}^\star\|_2^2$*, and* $k_i \in [k_{s_i}, k_{s_i+1})$ *be* $k_i = \max\{k : \operatorname{supp}(\boldsymbol{w}^{k-1}) \neq \operatorname{supp}(\boldsymbol{w}^\star)\}$ *with* (k_s) *the sequence of iterations defined in* (3)*.*

We can see from this theorem that after identification; the convergence rate does not depend on probability p. This means that communications become lessen with the same rate. On the other hand, as identification depends on this probability p, in practice, the selection of p should be a trade off between speed of identification and the size of sparsification.

4 Numerical experiments

In the previous sections we proved the convergence of `SODA` in the case where the mask is formed with a high probability p. In this section, we present numerical results providing empirical evidence on a faster execution of `SODA` with lower communication cost than if the mask is not used.

4.1 Experimental setup

In our experiments, we consider $\ell_1 - \ell_2$-regularized Logistic Regression surrogate loss that is common to many machine learning and signal processing applications and which can be minimized in a distributed way. With respect to our composite learning problem (1), that is :

$$\forall i \in \{1, \dots, M\}, f_i(\boldsymbol{w}) = \frac{1}{|\mathcal{S}_i|} \sum_{(\mathbf{x}_j, y_j) \in \mathcal{S}_i} \left[\log\left(1 + e^{-y_j \mathbf{x}_j^\top \boldsymbol{w}}\right) + \frac{\lambda_2}{2}\|\boldsymbol{w}\|_2^2\right] \tag{6}$$

where $\mathcal{S}_i = (\mathbf{x}_j, y_j)_{j \in \{1,\dots,|\mathcal{S}_j|} \in (\mathbb{R}^n \times \{-1,+1\})^{|\mathcal{S}_j|}$ is the sub-part of the training set stored in the worker machine $i \in \{1, \dots, M\}$.

We performed experiments on three publicly available datasets[1]. Each dataset is normalized by dividing each feature characteristic by the maximum of the absolute value in the column using the scikit-learn Transformer API.[2] In Table 1, we present some statistics for these datasets as well as the percentage of no-zero entries of the final parameter ($\operatorname{supp}(\boldsymbol{w}^\star)$).

For the communications between the master and the workers, we used the message passing interface for Python (MPI4py)[3]. We compared our approach

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

[2] https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.normalize.html

[3] https://mpi4py.readthedocs.io/en/stable/citing.html

Dataset	m	n	λ_1	λ_2	$\lvert\mathrm{supp}(\boldsymbol{w}^\star)\rvert$ in (%)
madelon	2000	500	2×10^{-2}	10^{-3}	7
real-sim	72309	20958	10^{-4}	10^{-3}	8.6
rcv1_train	20242	47236	10^{-4}	10^{-3}	4.1

Table 1. Statistics of datasets used in our experiments: λ_1, λ_2 are respectively the hyperparmeters corresponding to ℓ_1 and ℓ_2 regularization terms, and the percentage of non-zero entries of the final solution w.r.t. this selection.

SODA with its direct competitor DAve-PG [14] which was shown to outperform other state-of-the-art asynchronous distributed algorithms. For comparisons, we plot objective values as their relative distance to the optimum, referred to as suboptimality, with respect to time, and also with respect to iterations and the number of exchanges for different values of the probability p used in the mask. We also present the dependence of sparsity of iterates to the number iterates.

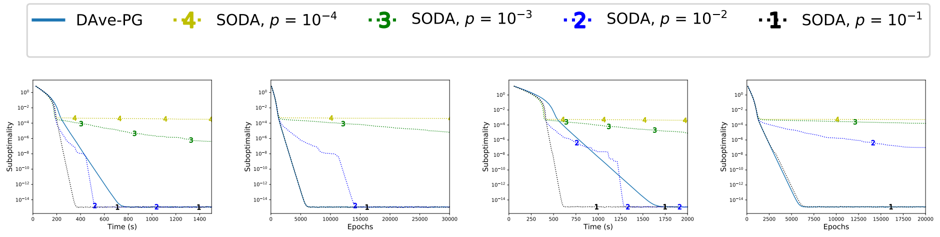

Fig. 1. Suboptimality versus time and epoch and the evolution of supp($\boldsymbol{w}^\star$) with respect to time for real-sim (left) and rcv1_train (right) datasets.

Speed of convergence. Figure 1 presents suboptimality versus time and epochs for the DAve-PG algorithm [14], and the SODA algortihm with four values of probability p, to form the mask with $M = 20$ workers on real-sim and (top) rcv1_train (down) datasets. To this end, the minimum of the loss function (1), using (6), is first obtained with a precision $\epsilon = 10^{-15}$. As it can be observed, for larger values of the probability p; SODA converges much faster than DAve-PG (up to 2 times faster for $p = 10^{-1}$ on both datasets). This is mainly because that SODA passes through the whole data (epochs) in less time than DAve-PG.

Cost of communication. We have computed the cost of communication, as the number of exchanges, between the master and the worker machines until convergence for different values of the probability p to form the mask. In the case where p is low, we know from the previous section that there is no guarantee that the algorithm SODA converges. However, note that as all workers are minimizing their local convex objectives, after one

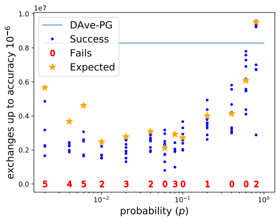

Fig. 2. madelon dataset

round of communication between the master and all workers, it is easy to detect when the global objective (1) does not decrease at the master level and the algorithm can be stopped and restarted in this case. In Figure 2 we plot the amount of exchanges between the master and the workers for different values of the probability p used to form the mask on the madelon dataset. For each value of p; we run the algorithm 10 times. Blue dots correspond to successful runs where the algorithm converged to the minimum of the objective (1) up to the precision 10^{-15}. Red numbers at the bottom of the figure mention the number of times when the algorithm diverged and expected amount of exchanges are shown by orange stars.

In addition, we plot the line (in cyan blue) for the number of exchanges of the `DAve-PG` algorithm [14]. As it can be seen, in mostly all the cases the `SODA` algorithm converges to the minimum of (1) with much fewer exchanges between the master and the workers than in the `DAve-PG` algorithm. For lower values of p, the number of times where the algorithm converges is low and for larger values of p; the expected number of exchanges tends to the one of the `DAve-PG` algorithm. This figure suggests that for this dataset the best compromise between the number of convergence and number of exchanges is reached for the values of $p \in [0.01, 0.6]$.

Evolution of sparsity. Let us now discuss the importance of sparsity, as the number of no-zero entries, of the final solution. Figure 3, shows the evolution of the percentage of no-zero entries of the parameter with respect to epochs on rcv1_train dataset for $M = 20$ workers and $\lambda_1 = 10^{-5}$. The sparsity of the solution increases over epochs for both `DAve-PG` and `SODA` algorithms. This is mainly due to the use of the ℓ_1 in both algorithms. This sparsification is accentuated for `SODA` by the use of the mask. From previous plots, we observed that for higher values of the probability p, the proposed algorithm converges faster to the minimum of the composite objective. From this figure, it comes out that the `SODA` algorithm is able to identify the same set of informative non-zero entries, than `DAve-PG`, at convergence for higher values of the probability p.

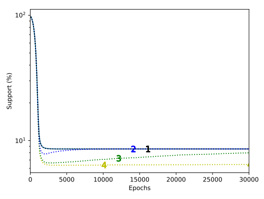

Fig. 3. rcv1_train dataset

5 Conclusion

In this paper, we proposed an asynchronous distributed learning algorithm with sparsification. The sparsification is induced through a mask that selects a subpart of the model parameters constituted with all non-zero entries and some others chosen randomly with a fixed probability p. We have analyzed the convergence property of the algorithm by showing that when p is moderately high the algorithm is ensured to converge for strongly convex composite objectives. In the case of small values of p, we have empirically shown on three benchmarks that when the algorithm converges, it reaches faster the minimum with much less communications between the master and the worker machines than if the mask is not used.

References

1. Bach, F., Jenatton, R., Mairal, J., Obozinski, G., et al.: Optimization with sparsity-inducing penalties. Foundations and Trends® in Machine Learning **4**(1), 1–106 (2012)
2. Boyd, S.P., Parikh, N., Chu, E., Peleato, B., Eckstein, J.: Distributed optimization and statistical learning via the alternating direction method of multipliers. Foundations and Trends in Machine Learning **3**(1), 1–122 (2011)
3. Candes, E.J., Wakin, M.B., Boyd, S.P.: Enhancing sparsity by reweighted l 1 minimization. Journal of Fourier analysis and applications **14**(5-6), 877–905 (2008)
4. Chen, J., Monga, R., Bengio, S., Jozefowicz, R.: Revisiting distributed synchronous sgd. In: International Conference on Learning Representations Workshop Track (2016), https://arxiv.org/abs/1604.00981
5. Fadili, J., Malick, J., Peyré, G.: Sensitivity analysis for mirror-stratifiable convex functions. SIAM Journal on Optimization **28**(4), 2975–3000 (2018)
6. Grishchenko, D., Iutzeler, F., Malick, J., Amini, M.R.: Asynchronous distributed learning with sparse communications and identification. arXiv preprint arXiv:1812.03871 (2018)
7. Hannah, R., Yin, W.: On unbounded delays in asynchronous parallel fixed-point algorithms. Journal of Scientific Computing pp. 1–28 (2016)
8. Konečnỳ, J., McMahan, H.B., Ramage, D., Richtárik, P.: Federated optimization: distributed machine learning for on-device intelligence. arXiv:1610.02527 (2016)
9. Kumar, V.: Introduction to Parallel Computing. Addison-Wesley Longman (2002)
10. Lee, S., Wright, S.J.: Manifold identification in dual averaging for regularized stochastic online learning. Journal of Machine Learning Research **13** (2012)
11. Lin, Y., Han, S., Mao, H., Wang, Y., Dally, W.J.: Deep gradient compression: Reducing the communication bandwidth for distributed training. arXiv preprint arXiv:1712.01887 (2017)
12. Ma, C., Jaggi, M., Curtis, F.E., Srebro, N., Takáč, M.: An accelerated communication-efficient primal-dual optimization framework for structured machine learning. Optimization Methods and Software pp. 1–25 (2019)
13. Mishchenko, K., Iutzeler, F., Malick, J.: A distributed flexible delay-tolerant proximal gradient algorithm. SIAM Journal on Optimization **30**(1), 933–959 (2020)
14. Mishchenko, K., Iutzeler, F., Malick, J., Amini, M.R.: A delay-tolerant proximal-gradient algorithm for distributed learning. In: Proceedings of the 35th International Conference on Machine Learning (ICML). vol. 80, pp. 3587–3595 (2018)
15. Nutini, J., Schmidt, M., Hare, W.: "active-set complexity" of proximal gradient: How long does it take to find the sparsity pattern? Optimization Letters **13**(4), 645–655 (2019)
16. Sun, T., Hannah, R., Yin, W.: Asynchronous coordinate descent under more realistic assumptions. In: Advances in Neural Information Processing Systems (2017)
17. Valiant, L.G.: A bridging model for parallel computation. Communications of the ACM **33**(8), 103–111 (1990)
18. Wangni, J., Wang, J., Liu, J., Zhang, T.: Gradient sparsification for communication-efficient distributed optimization. In: Advances in Neural Information Processing Systems. pp. 1306–1316 (2018)