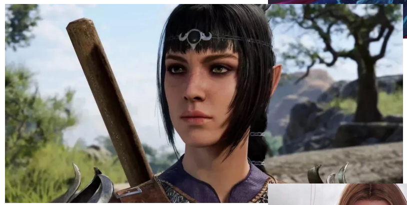
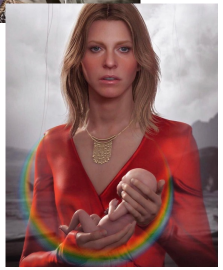
The First Optimal Parallel SGD
(in the Presence of Data, Compute and Communication Heterogeneity)

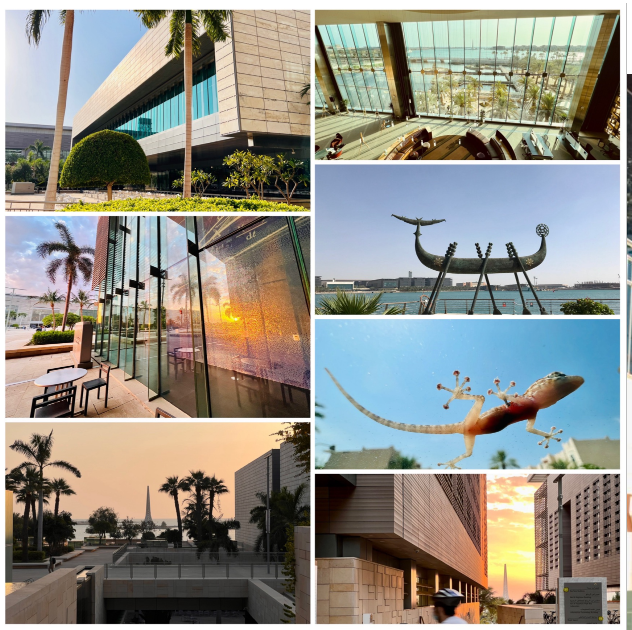
Peter Richtárik
King Abdullah University of Science and Technology
Kingdom of Saudi Arabia

3rd Workshop on Federated Learning for Computer Vision

in Conjunction with CVPR 2024
(6/17 All Day)

Optimization & Machine Learning Lab @ KAUST



Part 1
Federated Learning

Jakub Konečný

H Brendan McMahan

THE UNIVERSITY *of* EDINBURGH

Federated Learning was developed in 2015/2016 in a collaboration between the University of Edinburgh & Google

H. Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, Blaise Agüera y Arcas
Communication-Efficient Learning of Deep Networks from Decentralized Data
20th International Conference on Artificial Intelligence and Statistics (AISTATS), 2017

Keith Bonawitz et al
Practical Secure Aggregation for Federated Learning on User-Held Data
NIPS Private Multi-Party Machine Learning Workshop, 2016

Google AI Blog

The latest from Google Research

Federated Learning: Collaborative Machine Learning without Centralized Training Data

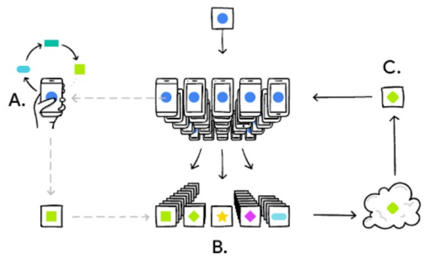
Thursday, April 6, 2017

Posted by Brendan McMahan and Daniel Ramage, Research Scientists

Standard machine learning approaches require centralizing the training data on one machine or in a datacenter. And Google has built one of the most secure and robust cloud infrastructures for processing this data to make our services better. Now for models trained from user interaction with mobile devices, we're introducing an additional approach: *Federated Learning*.

Federated Learning enables mobile phones to collaboratively learn a shared prediction model while keeping all the training data on device, decoupling the ability to do machine learning from the need to store the data in the cloud. This goes beyond the use of local models that make predictions on mobile devices (like the Mobile Vision API and On-Device Smart Reply) by bringing model *training* to the device as well.

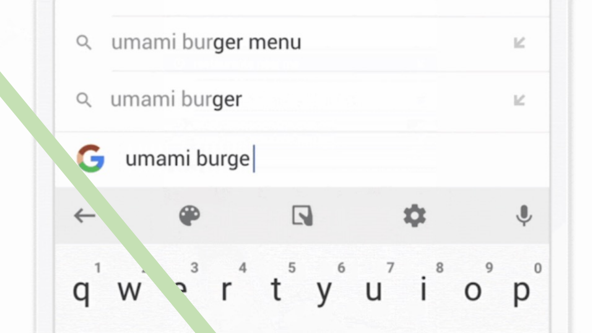
It works like this: your device downloads the current model, improves it by learning from data on your phone, and then summarizes the changes as a small focused update. Only this update to the model is sent to the cloud, using encrypted communication, where it is immediately averaged with other user updates to improve the shared model. All the training data remains on your device, and no individual updates are stored in the cloud.



Your phone personalizes the model locally, based on your usage (A). Many users' updates are aggregated (B) to form a consensus change (C) to the shared model, after which the procedure is repeated.

Federated Learning allows for smarter models, lower latency, and less power consumption, all while ensuring privacy. And this approach has another immediate benefit: in addition to providing an update to the shared model, the improved model on your phone can also be used immediately, powering experiences personalized by the way you use your phone.

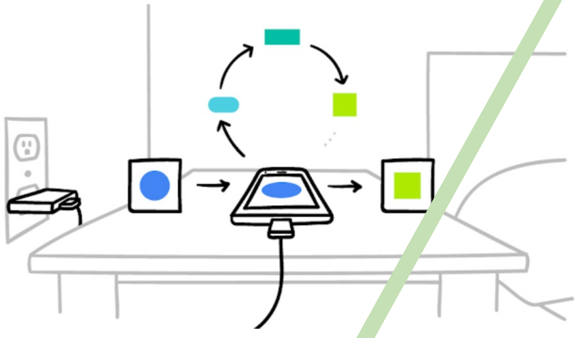
We're currently testing Federated Learning in Gboard on Android, the Google Keyboard. When Gboard shows a suggested query, your phone locally stores information about the current context and whether you clicked the suggestion. Federated Learning processes that history on-device to suggest improvements to the next iteration of Gboard's query suggestion model.



To make Federated Learning possible, we had to overcome many algorithmic and technical challenges. In a typical machine learning system, an optimization algorithm like Stochastic Gradient Descent (SGD) runs on a large dataset partitioned homogeneously across servers in the cloud. Such highly iterative algorithms require low-latency, high-throughput connections to the training data. But in the Federated Learning setting, the data is distributed across millions of devices in a highly uneven fashion. In addition, these devices have significantly higher-latency, lower-throughput connections and are only intermittently available for training.

These bandwidth and latency limitations motivate our Federated Averaging algorithm, which can train deep networks using 10-100x less communication compared to a naively federated version of SGD. The key idea is to use the powerful processors in modern mobile devices to compute higher quality updates than simple gradient steps. Since it takes fewer iterations of high-quality updates to produce a good model, training can use much less communication. As upload speeds are typically much slower than download speeds, we also developed a novel way to reduce upload communication costs up to another 100x by compressing updates using random rotations and quantization. While these approaches are focused on training deep networks, we've also designed algorithms for high-dimensional sparse convex models which excel on problems like click-through-rate prediction.

Deploying this technology to millions of heterogenous phones running Gboard requires a sophisticated technology stack. On device training uses a miniature version of TensorFlow. Careful scheduling ensures training happens only when the device is idle, plugged in, and on a free wireless connection, so there is no impact on the phone's performance.



Your phone participates in Federated Learning only when it won't negatively impact your experience.

The system then needs to communicate and aggregate the model updates in a secure, efficient, scalable, and fault-tolerant way. It's only the combination of research with this infrastructure that makes the benefits of Federated Learning possible.

Federated learning works without the need to store user data in the cloud, but we're not stopping there. We've developed a Secure Aggregation protocol that uses cryptographic techniques so a coordinating server can only decrypt the average update if 100s or 1000s of users have participated — no individual phone's update can be inspected before averaging. It's the first protocol of its kind that is practical for deep-network-sized problems and real-world connectivity constraints. We designed Federated Averaging so the coordinating server only needs the average update, which allows Secure Aggregation to be used; however the protocol is general and can be applied to other problems as well. We're working hard on a production implementation of this protocol and expect to deploy it for Federated Learning applications in the near future.

Our work has only scratched the surface of what is possible. Federated Learning can't solve all machine learning problems (for example, learning to recognize different dog breeds by training on carefully labeled examples), and for many other models the necessary training data is already stored in the cloud (like training spam filters for Gmail). So Google will continue to advance the state-of-the-art for cloud-based ML, but we are also committed to ongoing research to expand the range of problems we can solve with Federated Learning. Beyond Gboard query suggestions, for example, we hope to improve the language models that power your keyboard based on what you actually type on your phone (which can have a style all its own) and photo rankings based on what kinds of photos people look at, share, or delete.

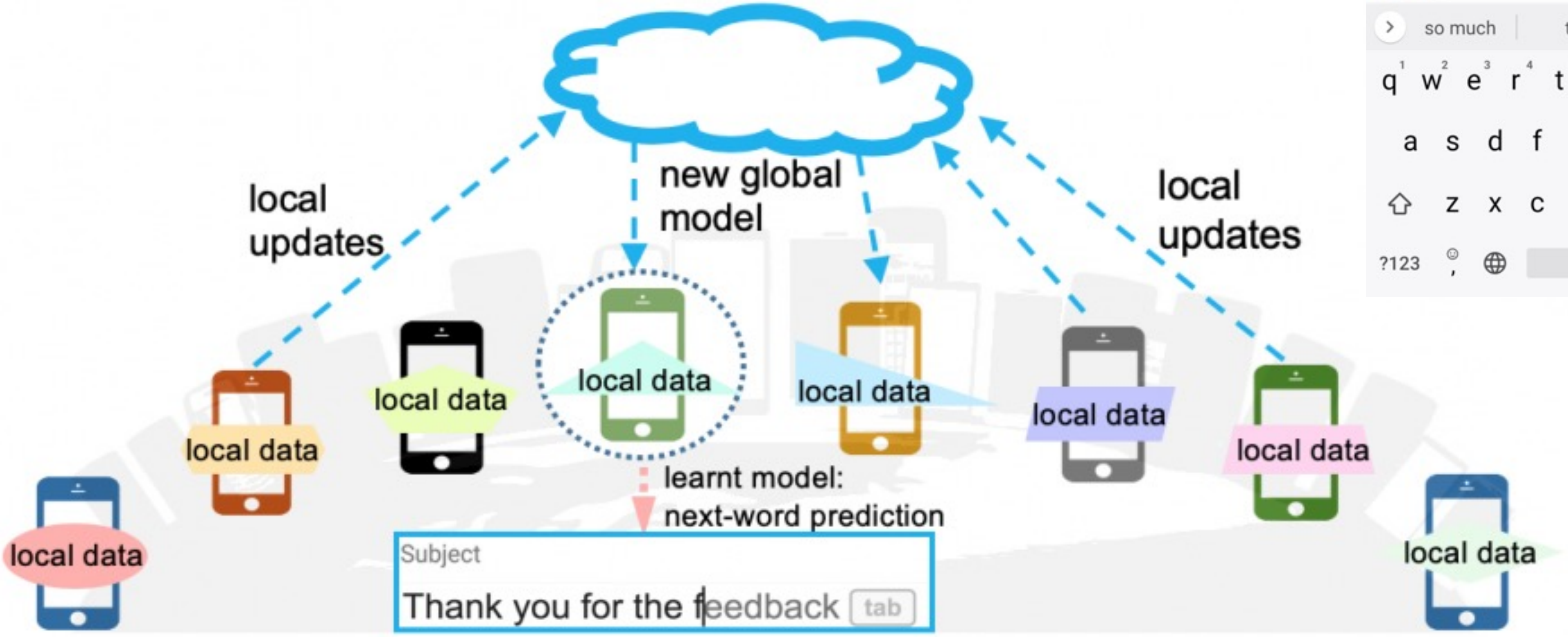
Applying Federated Learning requires machine learning practitioners to adopt new tools and a new way of thinking: model development, training, and evaluation with no direct access to or labeling of raw data, with communication cost as a limiting factor. We believe the user benefits of Federated Learning make tackling the technical challenges worthwhile, and are publishing our work with hopes of a widespread conversation within the machine learning community.

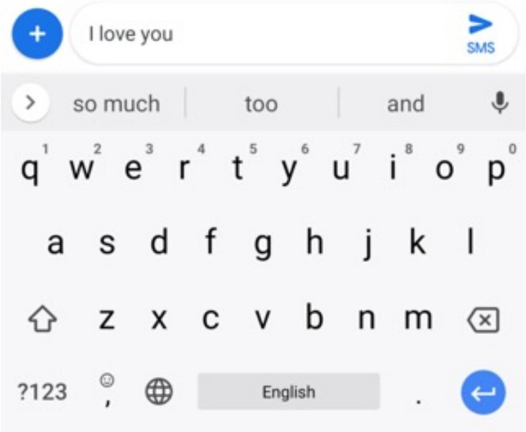
Jakub Konečný, H. Brendan McMahan, Felix X. Yu, Peter Richtárik, Ananda Theertha Suresh, Dave Bacon
Federated Learning: Strategies for Improving Communication Efficiency
NIPS Private Multi-Party Machine Learning Workshop, 2016

Jakub Konečný, H. Brendan McMahan, Daniel Ramage, Peter Richtárik
Federated Optimization: Distributed Machine Learning for On-Device Intelligence
arXiv:1610.02527, 2016

The First Federated Learning App: Next-Word Prediction

Federated Learning is collaborative machine learning from private data stored across a (large) number of clients/devices (e.g., hospitals, phones)





Google Scholar

Peter Richtarik

FOLLOWING

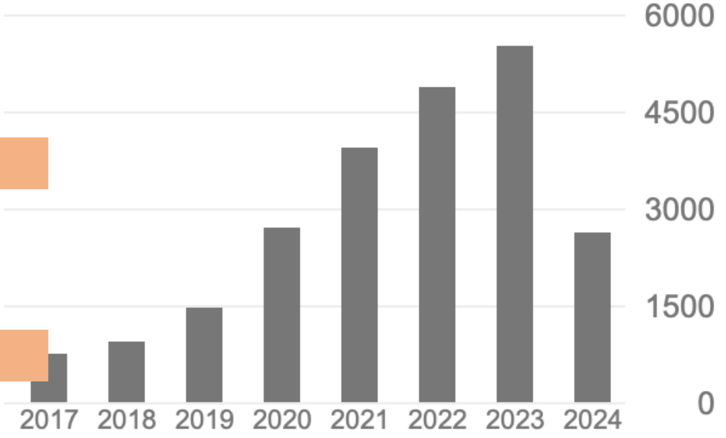
Professor, KAUST
Verified email at kaust.edu.sa - Homepage

optimization machine learning federated learning deep learning computer science

TITLE	CITED BY	YEAR
Federated learning: Strategies for improving communication efficiency J Konecný, HB McMahan, FX Yu, P Richtárik, AT Suresh, D Bacon arXiv preprint arXiv:1610.05492 8	3856	2016
Federated learning: Strategies for improving communication efficiency J Konečný, HB McMahan, FX Yu, P Richtárik, AT Suresh, D Bacon arXiv preprint arXiv:1610.05492	2716	2016
Federated optimization: Distributed machine learning for on-device intelligence J Konečný, HB McMahan, D Ramage, P Richtárik arXiv preprint arXiv:1610.02527	2091	2016
Iteration complexity of randomized block-coordinate descent methods for minimizing a composite function P Richtarik, M Takáč Mathematical Programming 144 (2), 1-38	860	2014

Cited by VIEW ALL

	All	Since 2019
Citations	24873	21252
h-index	69	63
i10-index	180	171



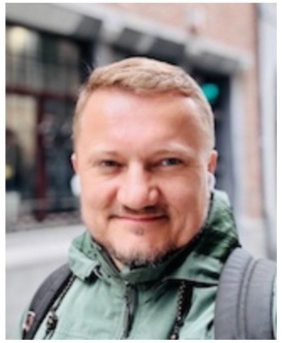
ublic access VIEW ALL

0 articles 43 articles

not available available

Based on funding mandates

My Team: 100+ Papers on Federated Learning



Peter Richtárik

Professor of Computer Science

KAUST

Address: Office 3145, Bldg 12, 4700 KAUST, Thuwal 23955-6900, Saudi Arabia
E-mail: peter.richtarik@kaust.edu.sa

dblp

All papers are listed below in reverse chronological order in which they appeared online.

Prepared in 2024

[258] Kai Yi, Timur Kharisov, Igor Sokolov, and Peter Richtárik
Cohort squeeze: Beyond a single communication round per cohort in cross-device federated learning
Federated Learning Paper
[arXiv] [method: SPPM-AS]

[257] Georg Meinhardt, Kai Yi, Laurent Condat, and Peter Richtárik
Prune at the clients, not the server: Accelerated sparse training in federated learning
Federated Learning Paper
[arXiv] [method: Sparse-ProxSkip]

[256] Avetik Karagulyan, Egor Shulgin, Abdurakhmon Sadiev, and Peter Richtárik
SPAM: Stochastic proximal point method with momentum variance reduction for non-convex cross-device federated learning
Federated Learning Paper
[arXiv] [method: SPAM]

Forbes

AI

The Next Generation Of Artificial Intelligence

Rob Toews Contributor
I write about the big picture of artificial intelligence.

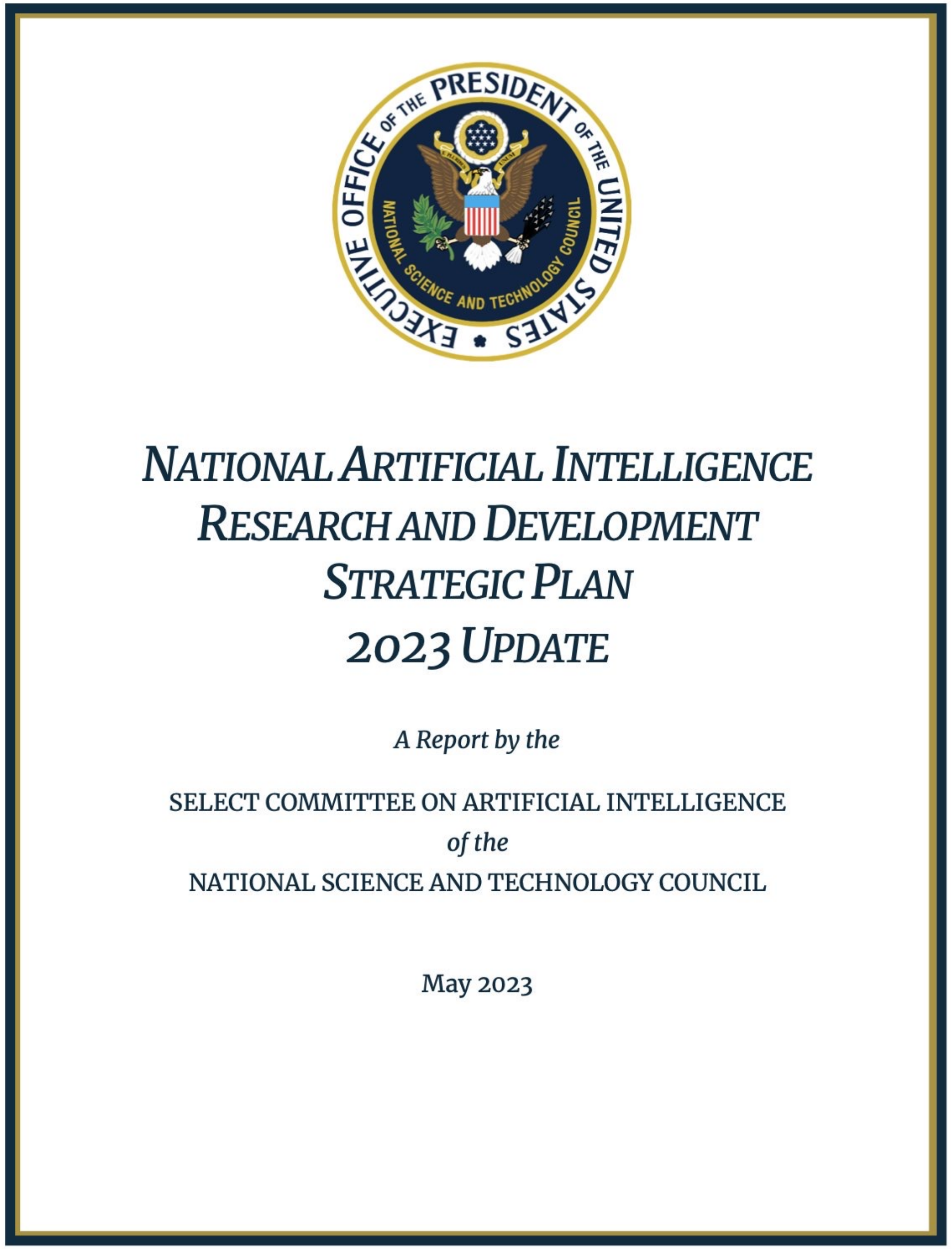
Follow

Oct 12, 2020, 09:22pm EDT

1. Unsupervised Learning
2. Federated Learning
3. Transformers
4. Neural Network Compression
5. Generative AI
6. “System 2” Reasoning

Google
OWKIN
Meta
FedML
NVIDIA
WeBank 微众银行
IBM
SAMSUNG
Baidu 百度

https://www.forbes.com/sites/robtoews/2020/10/12/the-next-generation-of-artificial-intelligence/?sh=4d14f60159eb
https://www.forbes.com/sites/robtoews/2020/10/29/the-next-generation-of-artificial-intelligence-part-2/?sh=e02f2567a304


NATIONAL ARTIFICIAL INTELLIGENCE RESEARCH AND DEVELOPMENT STRATEGIC PLAN 2023 UPDATE

A Report by the

SELECT COMMITTEE ON ARTIFICIAL INTELLIGENCE
of the
NATIONAL SCIENCE AND TECHNOLOGY COUNCIL

May 2023

The National Artificial Intelligence R&D Strategic Plan

Table of Contents

https://www.whitehouse.gov/wp-content/uploads/2023/05/National-Artificial-Intelligence-Research-and-Development-Strategic-Plan-2023-Update.pdf

Part 2
Introduction

Optimization Problem

$$\min_{x \in \mathbb{R}^d} f(x) \stackrel{\text{def}}{=} \frac{1}{n} \sum_{i=1}^{n} f_i(x)$$

parallel machines (n)

model parameters / features (d)

Loss on local data $\mathcal{D}_i$ stored on machine i

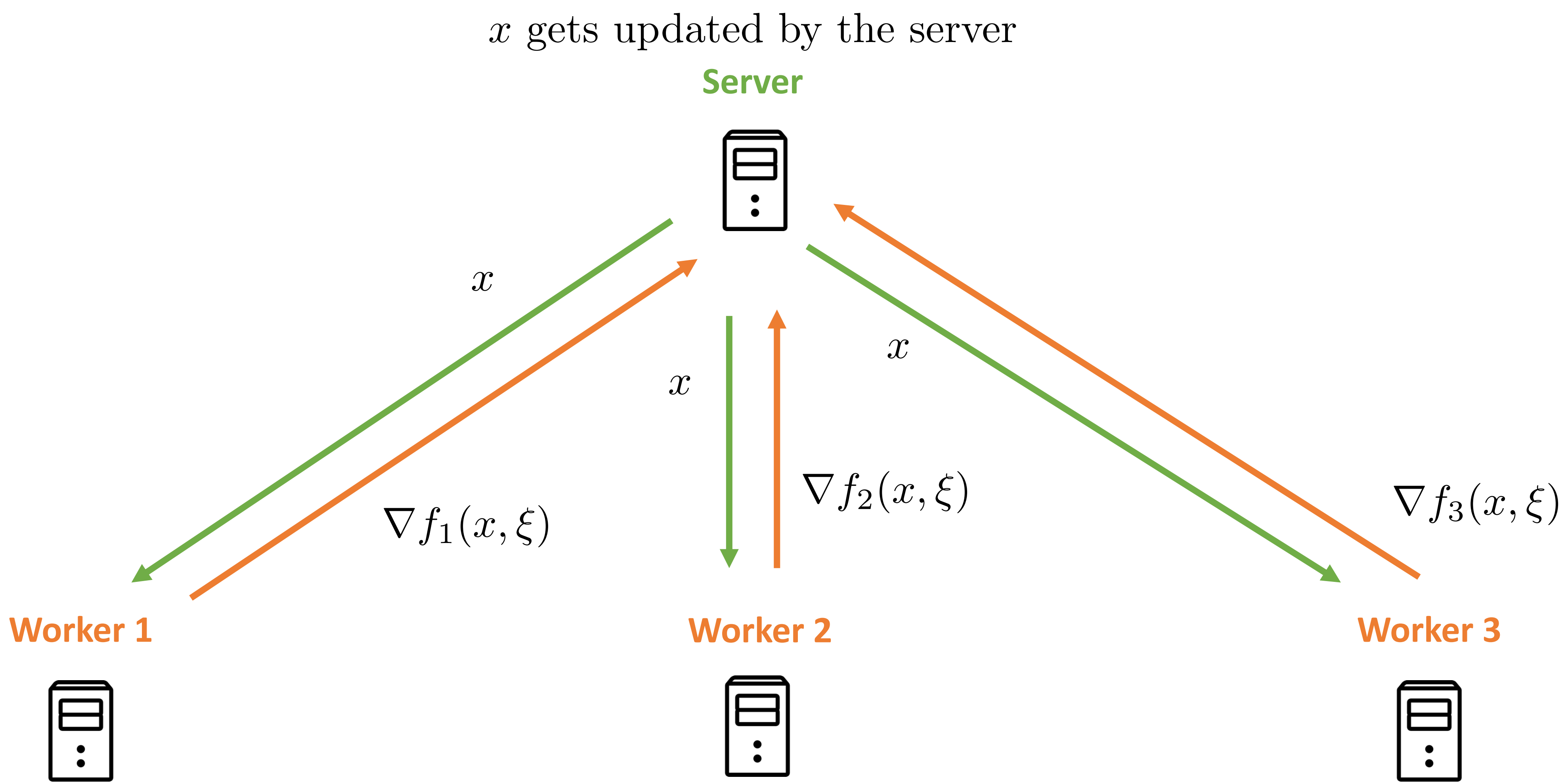
$$f_i(x) := \mathbb{E}_{\xi \sim \mathcal{D}_i} \left[f_i(x, \xi) \right]$$

! It takes τ_i seconds for worker i to compute $\nabla f_i(x, \xi)$, where $\xi \sim \mathcal{D}_i$ $\quad 0 < \tau_1 \leq \tau_2 \leq \cdots \leq \tau_n$

It takes θ_i seconds for worker i to communicate $g \in \mathbb{R}^d$ to the server

Find a (possibly random) vector $\hat{x} \in \mathbb{R}^d$ such that $\mathbb{E}\left[\|\nabla f(\hat{x})\|^2\right] \leq \varepsilon$

Parallel Computing Architecture



$f_1(x) := \mathbb{E}_{\xi \sim \mathcal{D}_1}\left[f_1(x, \xi)\right]$

$\nabla f_1(x, \xi)$ compute time $= \tau_1$ secs

$f_2(x) := \mathbb{E}_{\xi \sim \mathcal{D}_2}\left[f_2(x, \xi)\right]$

$\nabla f_2(x, \xi)$ compute time $= \tau_2$ secs

$f_3(x) := \mathbb{E}_{\xi \sim \mathcal{D}_3}\left[f_3(x, \xi)\right]$

$\nabla f_3(x, \xi)$ compute time $= \tau_3$ secs

Three Types of Heterogeneity

Data	data distributions $\mathcal{D}_1, \ldots, \mathcal{D}_n$ can be different
Compute	compute times $\tau_1, \ldots, \tau_n$ are nonzero and can be different
Communication	communication times $\theta_1, \ldots, \theta_n$ are nonzero and can be different

Typical Assumptions

1. $$\inf f \in \mathbb{R}$$

2. $$f_i(x) := \mathbb{E}_{\xi \sim \mathcal{D}_i}\left[f_i(x, \xi)\right]$$

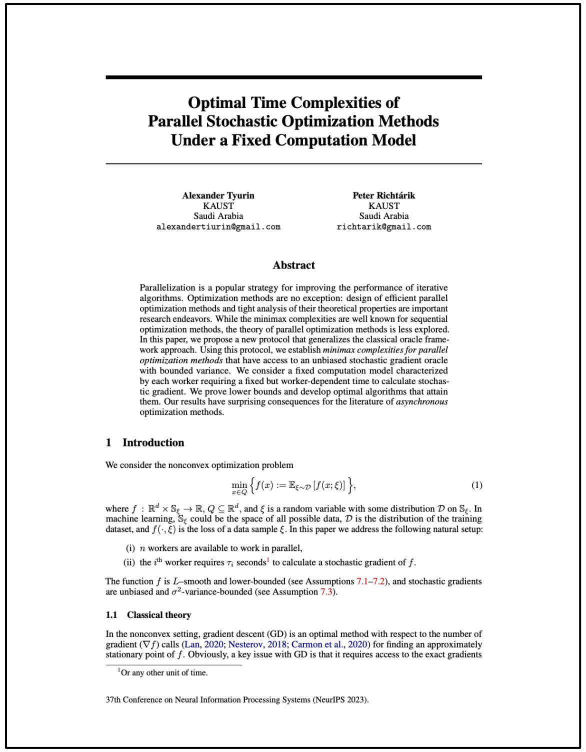
Gradient of local functions is Lipschitz:

$$\max_{i \in \{1,\ldots,n\}} \sup_{x \neq y} \frac{\|\nabla f_i(x) - \nabla f_i(y)\|}{\|x - y\|} \leq L$$

Stochastic gradients have bounded variance:

$$\max_{i \in \{1,\ldots,n\}} \sup_{x \in \mathbb{R}^d} \mathbb{E}_{\xi \sim \mathcal{D}_i}\left[\|\nabla f_i(x, \xi) - \mathbb{E}_{\xi \sim \mathcal{D}_i}\left[\nabla f_i(x, \xi)\right]\|^2\right] \leq \sigma^2$$

Our Papers on Optimal Parallel SGD

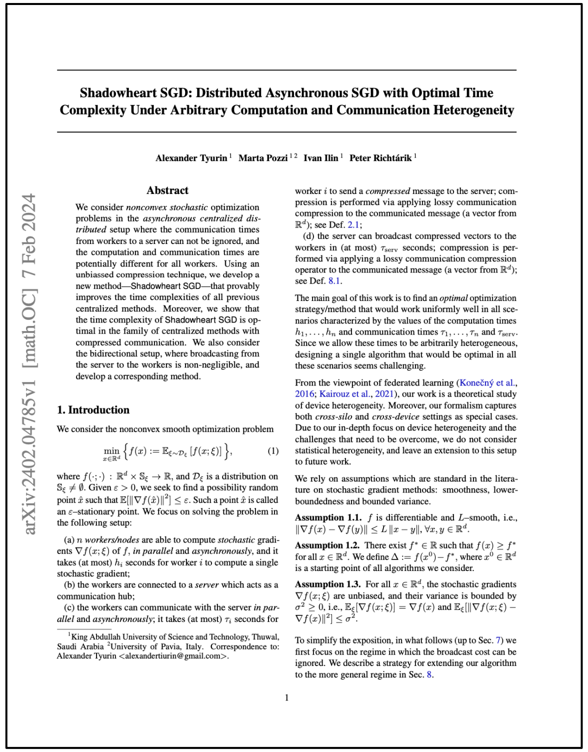


Optimal Time Complexities of Parallel Stochastic Optimization Methods Under a Fixed Computation Model

Alexander Tyurin, KAUST, Saudi Arabia, alexandertiurin@gmail.com
Peter Richtárik, KAUST, Saudi Arabia, richtarik@gmail.com

37th Conference on Neural Information Processing Systems (NeurIPS 2023).

5/2023

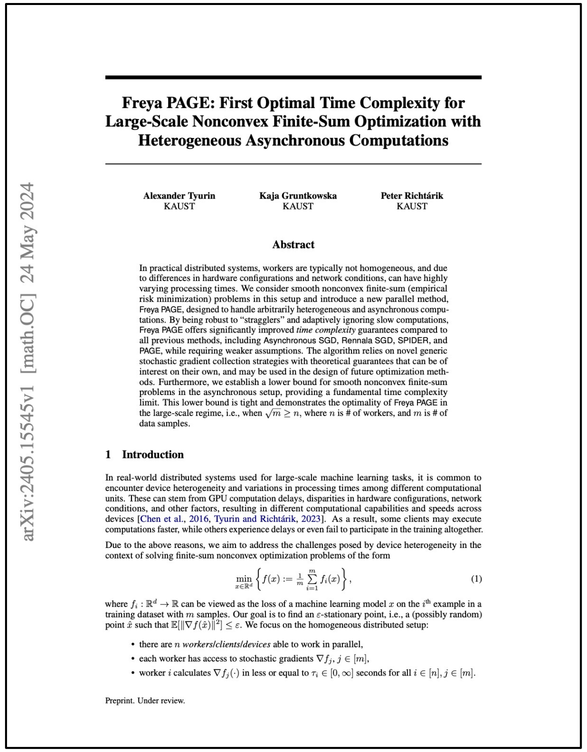


arXiv:2402.04785v1 [math.OC] 7 Feb 2024

Shadowheart SGD: Distributed Asynchronous SGD with Optimal Time Complexity Under Arbitrary Computation and Communication Heterogeneity

Alexander Tyurin[1] Marta Pozzi[1 2] Ivan Ilin[1] Peter Richtárik[1]

2/2024



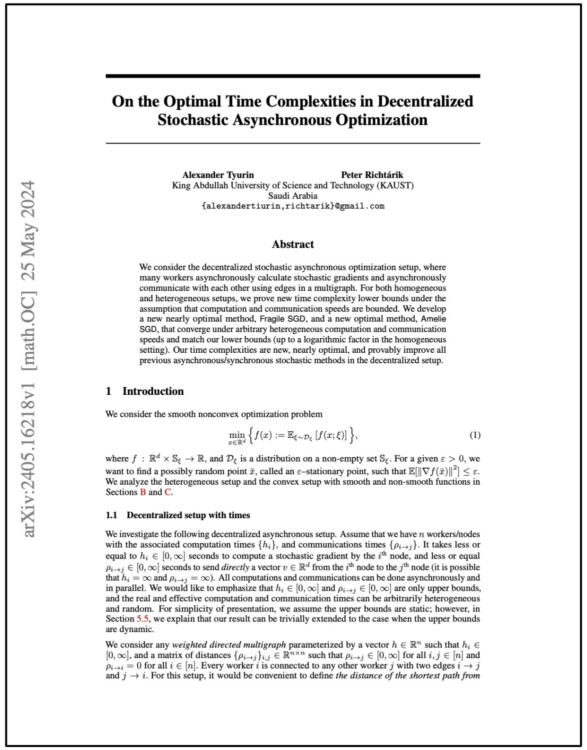
arXiv:2405.15545v1 [math.OC] 24 May 2024

Freya PAGE: First Optimal Time Complexity for Large-Scale Nonconvex Finite-Sum Optimization with Heterogeneous Asynchronous Computations

Alexander Tyurin, KAUST
Kaja Gruntkowska, KAUST
Peter Richtárik, KAUST

Preprint. Under review.

5/2024



arXiv:2405.16218v1 [math.OC] 25 May 2024

On the Optimal Time Complexities in Decentralized Stochastic Asynchronous Optimization

Alexander Tyurin, Peter Richtárik
King Abdullah University of Science and Technology (KAUST)
Saudi Arabia
{alexandertiurin,richtarik}@gmail.com

5/2024

Our Papers

First optimal parallel SGD under…

5/2023

Rennala SGD
Malenia SGD
Acc. Rennala SGD

Alexander Tyurin and P.R.
Optimal time complexities of parallel stochastic optimization methods under a fixed computation model
NeurIPS 2023

*… **computation** (and/or data) **heterogeneity***

2/2024

Shadowheart SGD

Alexander Tyurin, Marta Pozzi, Ivan Ilin and P.R.
Shadowheart SGD: Distributed asynchronous SGD with optimal time complexity under arbitrary computation and communication heterogeneity
arXiv:2402.04785, 2024

*… **communication** (and computation) **heterogeneity***

[Rennala SGD as a special case]

5/2024

Freya PAGE
Freya SGD

Alexander Tyurin, Kaja Gruntkowska, and P.R.
Freya PAGE: First optimal time complexity for large-scale nonconvex finite-sum optimization with heterogeneous asynchronous computations
arXiv:2405.1554, 2024

*… computation heterogeneity for **finite-sum** problems*

in the large-scale regime: $m \geq n^2$

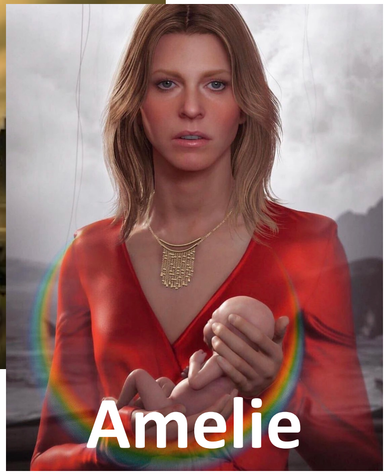
5/2024

Fragile SGD, Amelie SGD
+ accelerated variants

Alexander Tyurin and P.R.
On the optimal time complexities in decentralized stochastic asynchronous optimization
arXiv:2405.16218, 2024

*… computation and communication heterogeneity in the **decentralized setup***

Peter, What About the Weird Algorithm Names?



Rennala, Queen of the Full Moon is a Legend Boss in Elden Ring. Though not a demigod, Rennala is one of the shardbearers who resides in the Academy of Raya Lucaria. Rennala is a powerful sorceress, head of the Carian Royal family, and erstwhile leader of the Academy.

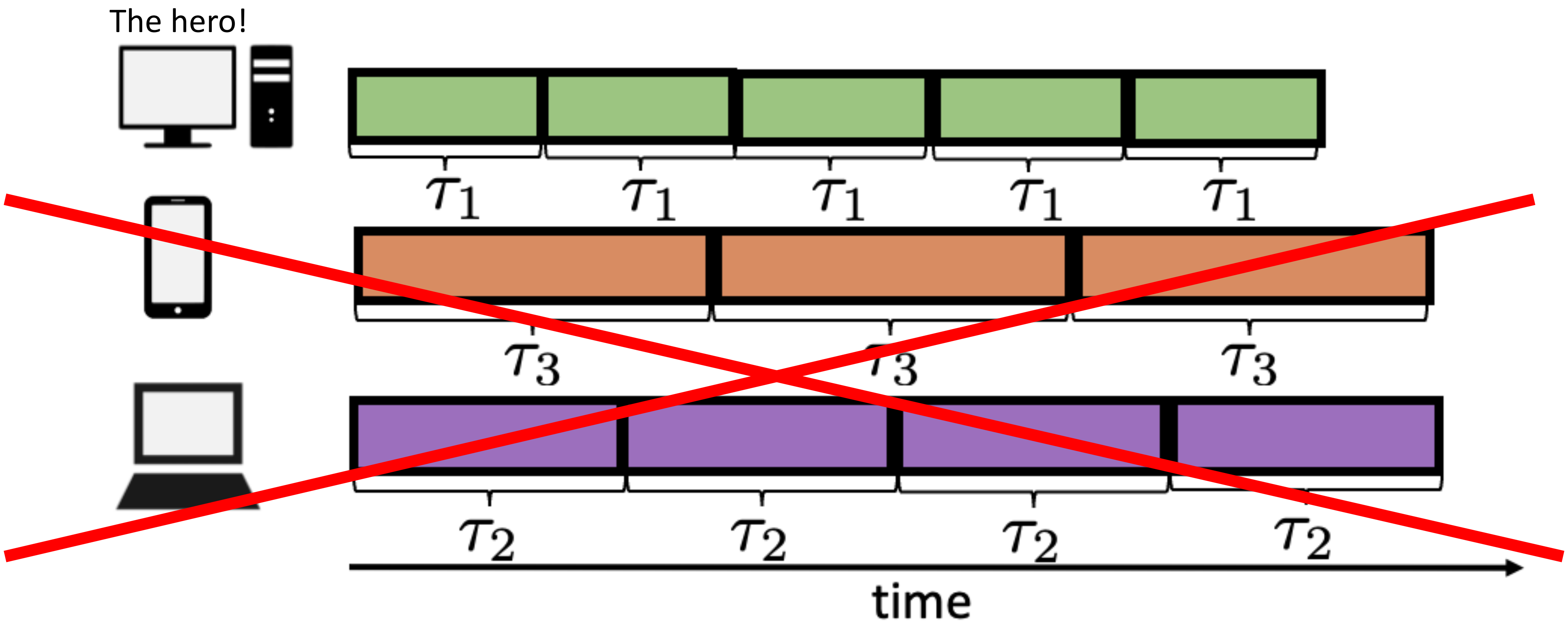
Optimal Parallel Stochastic Gradient Methods

	Data Heterogeneity ($\mathcal{D}_i$ different)	Compute Heterogeneity (τ_i different)	Communication Heterogeneity (θ_i different)	Smooth Nonconvex	Smooth Convex	Infinite / Finite Sum?	Supports Decentralized Setup?	Optimal Time Complexity?
Rennala SGD Tyurin & R (NeurIPS '23)	✗	✓	0	✓		**Inf**	✗	✓
Malenia SGD Tyurin & R (NeurIPS '23)	✓	✓	0	✓		**Inf**	✗	✓
Accelerated Rennala SGD Tyurin & R (NeurIPS '23)	✗	✓	0		✓	**Inf**	✗	✓
Shadowheart SGD Tyurin, Pozzi, Ilin & R '24	✗	✓	✓	✓		**Inf**	✗	✓
Freya PAGE Tyurin, Gruntkowska & R '24	✗	✓	0	✓		**Finite**	✗	✓ big data regime
Freya SGD Tyurin, Gruntkowska & R '24	✗	✓	0	✓		**Finite**	✗	✗
Fragile SGD Tyurin & R '24	✗	✓	✓	✓		**Inf**	✓	nearly
Amelie SGD Tyurin & R '24	✓	✓	✓	✓		**Inf**	✓	✓

Part 3
Previous Approaches to Parallelizing SGD

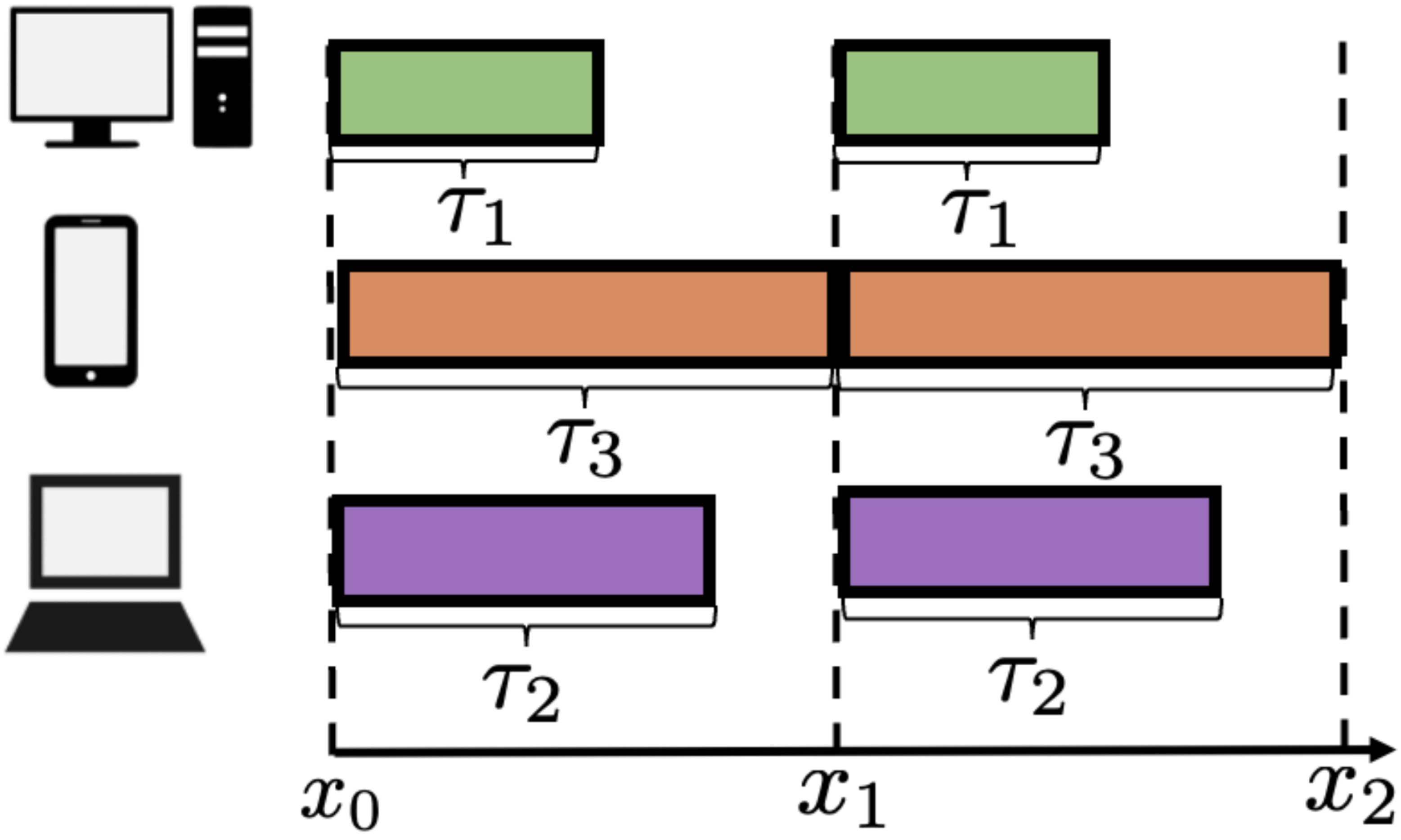
Hero SGD

Algorithmic idea: The fastest worker does it all!



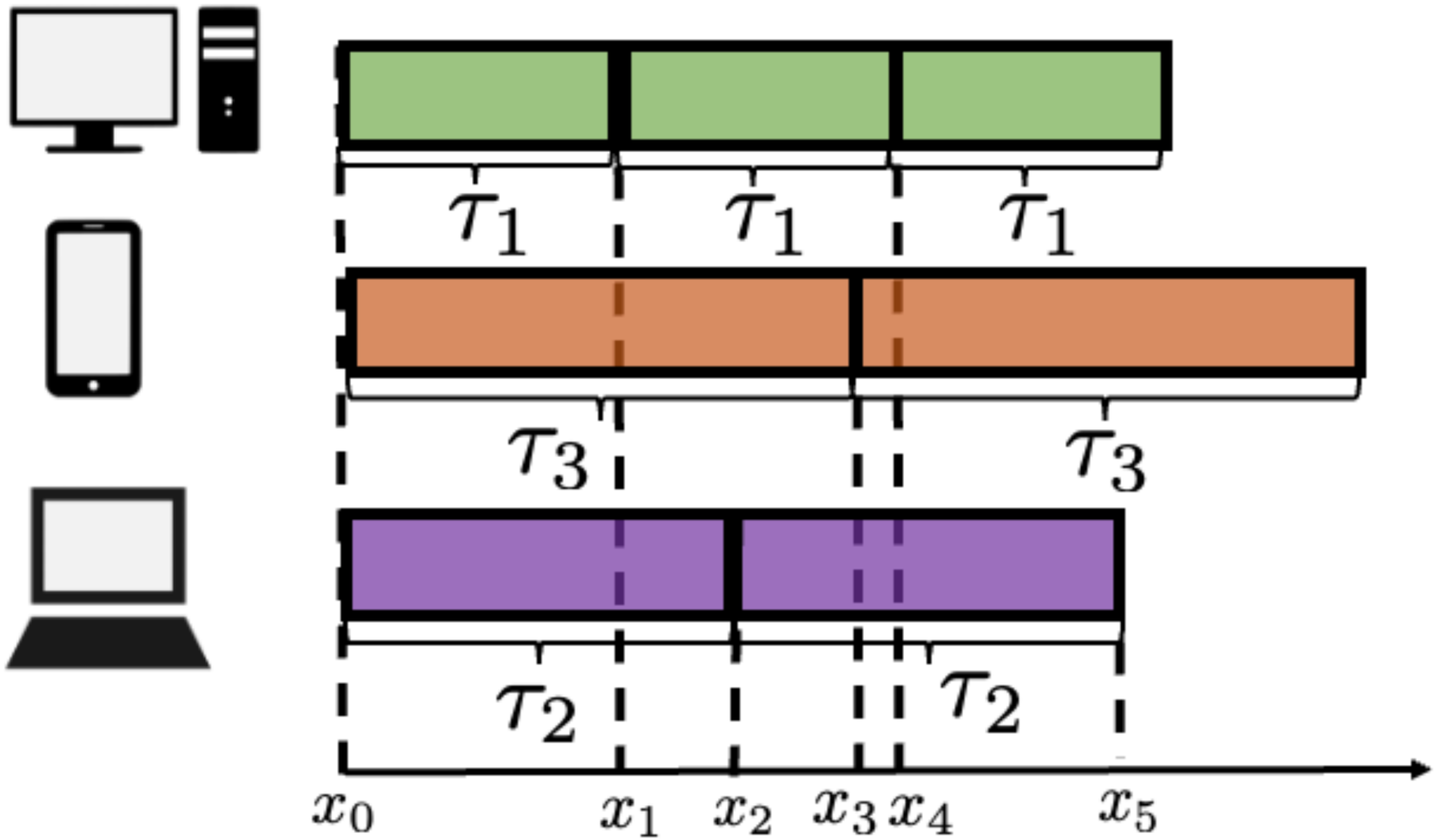
(Fair) Minibatch SGD

Algorithmic idea: Each worker does one job only!



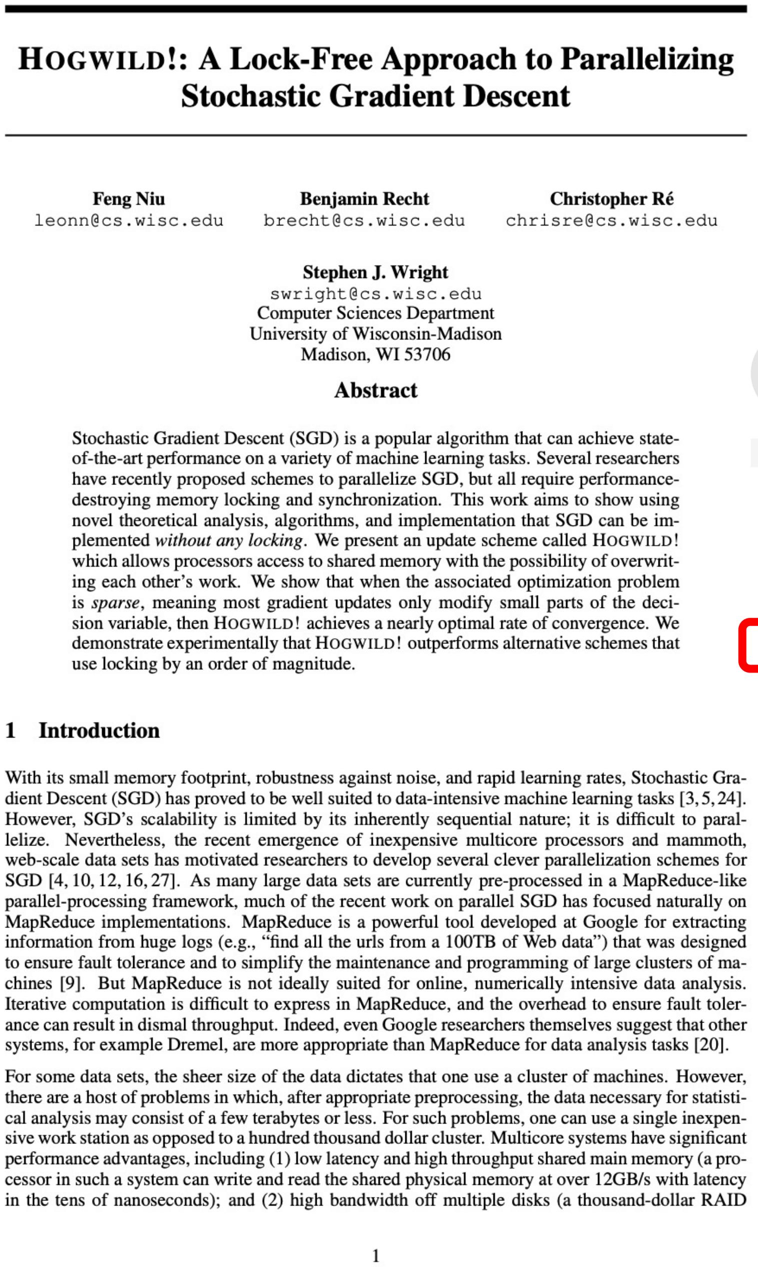
Asynchronous SGD

Algorithmic idea: All workers are slaves and useful



published in NIPS 2011

NeurIPS 2020 Test of Time Award



HOGWILD!: A Lock-Free Approach to Parallelizing Stochastic Gradient Descent

Feng Niu leonn@cs.wisc.edu
Benjamin Recht brecht@cs.wisc.edu
Christopher Ré chrisre@cs.wisc.edu

Stephen J. Wright
swright@cs.wisc.edu
Computer Sciences Department
University of Wisconsin-Madison
Madison, WI 53706

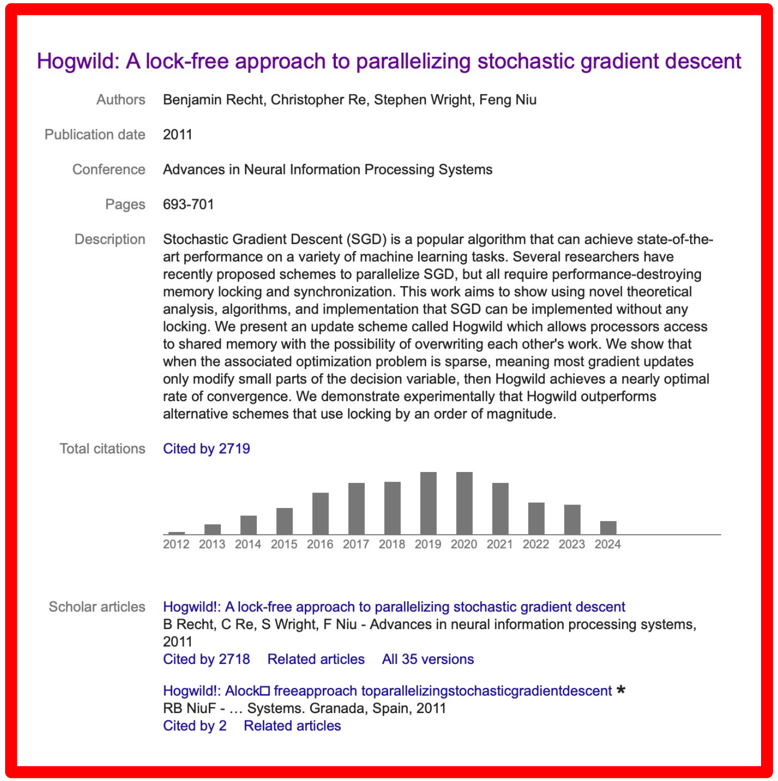
Abstract

Stochastic Gradient Descent (SGD) is a popular algorithm that can achieve state-of-the-art performance on a variety of machine learning tasks. Several researchers have recently proposed schemes to parallelize SGD, but all require performance-destroying memory locking and synchronization. This work aims to show using novel theoretical analysis, algorithms, and implementation that SGD can be implemented *without any locking*. We present an update scheme called HOGWILD! which allows processors access to shared memory with the possibility of overwriting each other's work. We show that when the associated optimization problem is *sparse*, meaning most gradient updates only modify small parts of the decision variable, then HOGWILD! achieves a nearly optimal rate of convergence. We demonstrate experimentally that HOGWILD! outperforms alternative schemes that use locking by an order of magnitude.

1 Introduction

With its small memory footprint, robustness against noise, and rapid learning rates, Stochastic Gradient Descent (SGD) has proved to be well suited to data-intensive machine learning tasks [3, 5, 24]. However, SGD's scalability is limited by its inherently sequential nature; it is difficult to parallelize. Nevertheless, the recent emergence of inexpensive multicore processors and mammoth, web-scale data sets has motivated researchers to develop several clever parallelization schemes for SGD [4, 10, 12, 16, 27]. As many large data sets are currently pre-processed in a MapReduce-like parallel-processing framework, much of the recent work on parallel SGD has focused naturally on MapReduce implementations. MapReduce is a powerful tool developed at Google for extracting information from huge logs (e.g., "find all the urls from a 100TB of Web data") that was designed to ensure fault tolerance and to simplify the maintenance and programming of large clusters of machines [9]. But MapReduce is not ideally suited for online, numerically intensive data analysis. Iterative computation is difficult to express in MapReduce, and the overhead to ensure fault tolerance can result in dismal throughput. Indeed, even Google researchers themselves suggest that other systems, for example Dremel, are more appropriate than MapReduce for data analysis tasks [20].

For some data sets, the sheer size of the data dictates that one use a cluster of machines. However, there are a host of problems in which, after appropriate preprocessing, the data necessary for statistical analysis may consist of a few terabytes or less. For such problems, one can use a single inexpensive work station as opposed to a hundred thousand dollar cluster. Multicore systems have significant performance advantages, including (1) low latency and high throughput shared main memory (a processor in such a system can write and read the shared physical memory at over 12GB/s with latency in the tens of nanoseconds); and (2) high bandwidth off multiple disks (a thousand-dollar RAID



Our Inspiration: Two Beautiful Papers

Asynchronous SGD Beats Minibatch SGD Under Arbitrary Delays

Konstantin Mishchenko Francis Bach Mathieu Even Blake Woodworth

DI ENS, Ecole normale supérieure,
Université PSL, CNRS, INRIA
75005 Paris, France

Abstract

The existing analysis of asynchronous stochastic gradient descent (SGD) degrades dramatically when any delay is large, giving the impression that performance depends primarily on the delay. On the contrary, we prove much better guarantees for the same asynchronous SGD algorithm regardless of the delays in the gradients, depending instead just on the number of parallel devices used to implement the algorithm. Our guarantees are strictly better than the existing analyses, and we also argue that asynchronous SGD outperforms synchronous minibatch SGD in the settings we consider. For our analysis, we introduce a novel recursion based on "virtual iterates" and delay-adaptive stepsizes, which allow us to derive state-of-the-art guarantees for both convex and non-convex objectives.

1 Introduction

We consider solving stochastic optimization problems of the form

$$\min_{\mathbf{x}\in\mathbb{R}^d} \{F(\mathbf{x}) := \mathbb{E}_{\xi\sim\mathcal{D}} f(\mathbf{x};\xi)\}, \quad (1)$$

which includes machine learning (ML) training objectives, where $f(\mathbf{x};\xi)$ represents the loss of a model parameterized by $\mathbf{x}$ on the datum ξ. Depending on the application, $\mathcal{D}$ could represent a finite dataset of size n or a population distribution. In recent years, such stochastic optimization problems have continued to grow rapidly in size, both in terms of the dimension d of the optimization variable—i.e., the number of model parameters in ML—and in terms of the quantity of data—i.e., the number of samples $\xi_1, \ldots, \xi_n \sim \mathcal{D}$ being used. With d and n regularly reaching the tens or hundreds of billions, it is increasingly necessary to use parallel optimization algorithms to handle the large scale and to benefit from data stored on different machines.

There are many ways of employing parallelism to solve (1), but the most popular approaches in practice are first-order methods based on stochastic gradient descent (SGD). At each iteration, SGD employs stochastic estimates of ∇F to update the parameters as $\mathbf{x}_k = \mathbf{x}_{k-1} - \gamma_k \nabla f(\mathbf{x}_{k-1};\xi_{k-1})$ for an i.i.d. sample $\xi_{k-1} \sim \mathcal{D}$. Given M machines capable of computing these stochastic gradient estimates $\nabla f(\mathbf{x};\xi)$ in parallel, one approach to parallelizing SGD is what we call "Minibatch SGD." This refers to a synchronous, parallel algorithm that dispatches the current parameters $\mathbf{x}_{k-1}$ to each of the M machines, waits while they compute and communicate back their gradient estimates $\mathbf{g}^1_{k-1}, \ldots, \mathbf{g}^M_{k-1}$, and then takes a minibatch SGD step $\mathbf{x}_k = \mathbf{x}_{k-1} - \gamma_k \cdot \frac{1}{M}\sum_{m=1}^M \mathbf{g}^m_{k-1}$. This is a natural idea with long history [16, 18, 55] and it is a commonly used in practice [e.g., 22]. However, since Minibatch SGD waits for all M of the machines to finish computing their gradient estimates before updating, it proceeds only at the speed of the *slowest* machine.

There are several possible sources of delays: nodes may have heterogeneous hardware with different computational throughputs [23, 25], network latency can slow the communication of gradients, and

36th Conference on Neural Information Processing Systems (NeurIPS 2022).

arXiv: June 15, 2022

Sharper Convergence Guarantees for Asynchronous SGD for Distributed and Federated Learning

Anastasia Koloskova, EPFL, anastasia.koloskova@epfl.ch
Sebastian U. Stich, CISPA*, stich@cispa.de
Martin Jaggi, EPFL, martin.jaggi@epfl.ch

Abstract

We study the asynchronous stochastic gradient descent algorithm for distributed training over n workers which have varying computation and communication frequency over time. In this algorithm, workers compute stochastic gradients in parallel at their own pace and return those to the server without any synchronization. Existing convergence rates for this algorithm for non-convex smooth objectives depend on the maximum gradient delay $\tau_{\max}$ and show that an ε-stationary point is reached after $\mathcal{O}(\sigma^2\varepsilon^{-2} + \tau_{\max}\varepsilon^{-1})$ iterations, where σ denotes the variance of stochastic gradients.

In this work we obtain (i) a tighter convergence rate of $\mathcal{O}(\sigma^2\varepsilon^{-2} + \sqrt{\tau_{\max}\tau_{avg}}\varepsilon^{-1})$ *without any change in the algorithm*, where τ_{avg} is the average delay, which can be significantly smaller than $\tau_{\max}$. We also provide (ii) a simple delay-adaptive learning rate scheme, under which asynchronous SGD achieves a convergence rate of $\mathcal{O}(\sigma^2\varepsilon^{-2} + \tau_{avg}\varepsilon^{-1})$, and does not require any extra hyperparameter tuning nor extra communications. Our result allows to show *for the first time* that asynchronous SGD is *always faster* than mini-batch SGD. In addition, (iii) we consider the case of heterogeneous functions motivated by federated learning applications and improve the convergence rate by proving a weaker dependence on the maximum delay compared to prior works. In particular, we show that the heterogeneity term in convergence rate is only affected by the average delay within each worker.

1 Introduction

The stochastic gradient descent (SGD) algorithm [43, 13] and its variants (momentum SGD, Adam, etc.) form the foundation of modern machine learning and frequently achieve state of the art results. With recent growth in the size of models and available training data, parallel and distributed versions of SGD are becoming increasingly important [57, 17, 16]. Without those, modern state-of-the art language models [44], generative models [40, 41], and many others [50] would not be possible. In the distributed setting, also known as data-parallel training, optimization is distributed over many compute devices working in parallel (e.g. cores, or GPUs on a cluster) in order to speed up training. Every worker computes gradients on a subset of the training data, and the resulting gradients are aggregated (averaged) on a server.

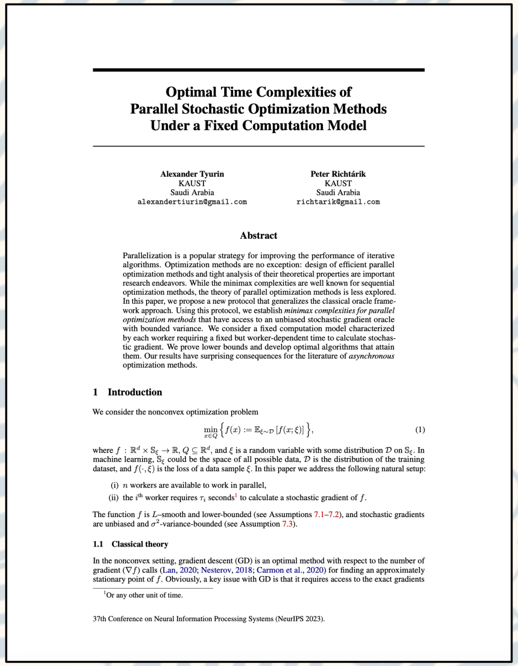
The same type of SGD variants also form the core algorithms for federated learning applications [34, 24] where the training process is naturally distributed over many user devices, or clients, that keep their local data private, and only transfer (e.g. encrypted or differentially private) gradients to the server.

A rich literature exists on the convergence theory of above mentioned parallel SGD methods, see e.g. [17, 13] and references therein. Plain parallel SGD still faces many challenges in practice, motivat-

*CISPA Helmholtz Center for Information Security

36th Conference on Neural Information Processing Systems (NeurIPS 2022).

arXiv: June 16, 2022


Optimal Time Complexities of
Parallel Stochastic Optimization Methods
Under a Fixed Computation Model

Alexander Tyurin
KAUST
Saudi Arabia
alexandertiurin@gmail.com

Peter Richtárik
KAUST
Saudi Arabia
richtarik@gmail.com

Abstract

Parallelization is a popular strategy for improving the performance of iterative algorithms. Optimization methods are no exception: design of efficient parallel optimization methods and tight analysis of their theoretical properties are important research endeavors. While the minimax complexities are well known for sequential optimization methods, the theory of parallel optimization methods is less explored. In this paper, we propose a new protocol that generalizes the classical oracle framework approach. Using this protocol, we establish *minimax complexities for parallel optimization methods* that have access to an unbiased stochastic gradient oracle with bounded variance. We consider a fixed computation model characterized by each worker requiring a fixed but worker-dependent time to calculate stochastic gradient. We prove lower bounds and develop optimal algorithms that attain them. Our results have surprising consequences for the literature of *asynchronous* optimization methods.

1 Introduction

We consider the nonconvex optimization problem

$$\min_{x \in Q} \left\{ f(x) := \mathbb{E}_{\xi \sim \mathcal{D}} \left[f(x;\xi) \right] \right\}, \quad (1)$$

where $f : \mathbb{R}^d \times \mathbb{S}_\xi \to \mathbb{R}$, $Q \subseteq \mathbb{R}^d$, and ξ is a random variable with some distribution $\mathcal{D}$ on $\mathbb{S}_\xi$. In machine learning, $\mathbb{S}_\xi$ could be the space of all possible data, $\mathcal{D}$ is the distribution of the training dataset, and $f(\cdot,\xi)$ is the loss of a data sample ξ. In this paper we address the following natural setup:

(i) n workers are available to work in parallel,
(ii) the i^{th} worker requires τ_i seconds[1] to calculate a stochastic gradient of f.

The function f is L–smooth and lower-bounded (see Assumptions 7.1–7.2), and stochastic gradients are unbiased and σ^2-variance-bounded (see Assumption 7.3).

1.1 Classical theory

In the nonconvex setting, gradient descent (GD) is an optimal method with respect to the number of gradient (∇f) calls (Lan, 2020; Nesterov, 2018; Carmon et al., 2020) for finding an approximately stationary point of f. Obviously, a key issue with GD is that it requires access to the exact gradients

[1] Or any other unit of time.

37th Conference on Neural Information Processing Systems (NeurIPS 2023).

Part 4
Rennala SGD

Alexander Tyurin and P.R.
Optimal time complexities of parallel stochastic optimization methods under a fixed computation model
NeurIPS 2023

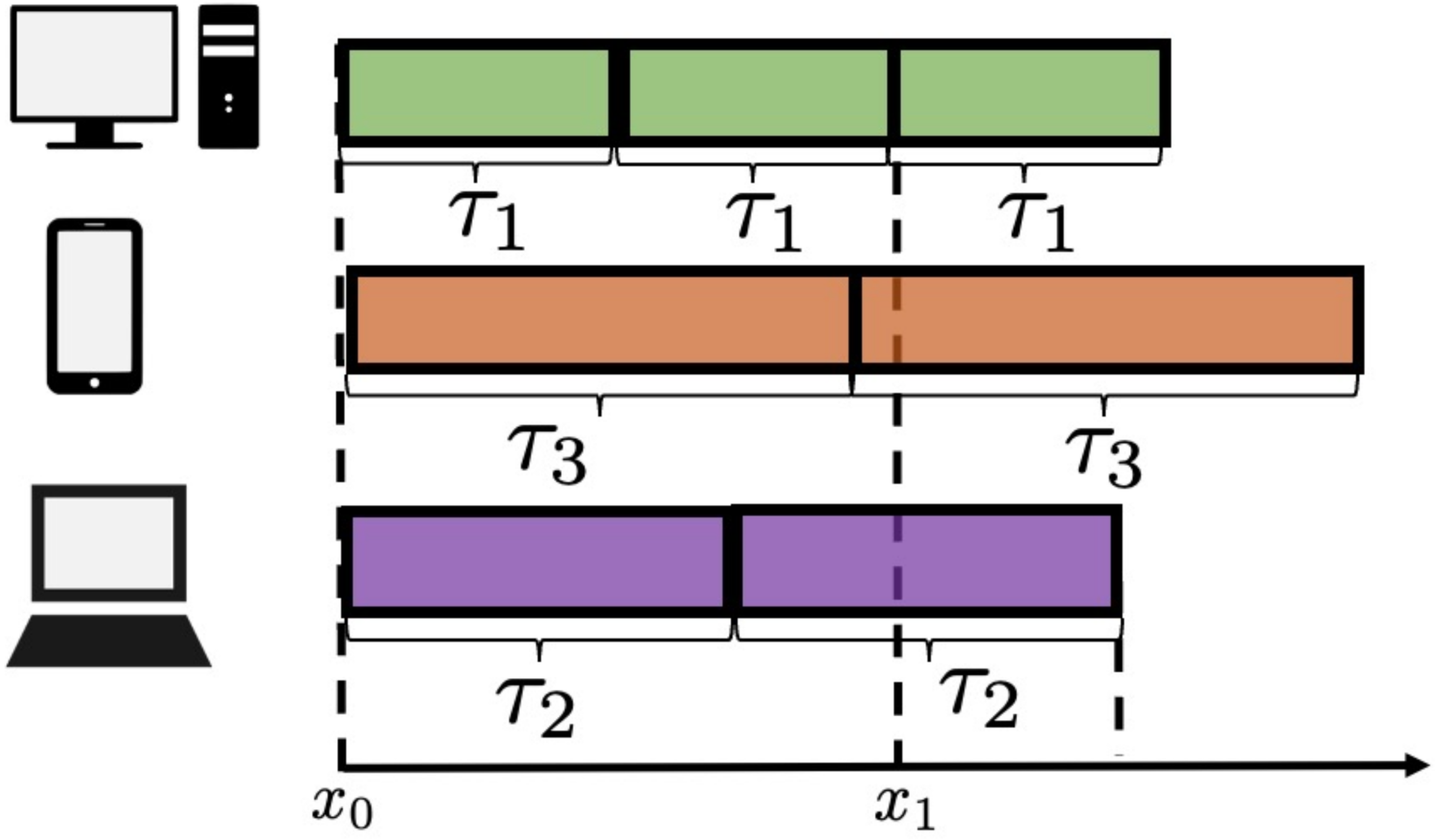
Setup

Optimal Parallel Stochastic Gradient Methods

	Data Heterogeneity ($\mathcal{D}_i$ different)	**Compute Heterogeneity** (τ_i different)	**Communication Heterogeneity** (θ_i different)	**Smooth Nonconvex**	**Smooth Convex**	**Infinite / Finite Sum?**	**Supports Decentralized Setup?**	**Optimal Time Complexity?**
Rennala SGD Tyurin & R (NeurIPS '23)	✗	✓	0	✓		**Inf**	✗	✓
Malenia SGD Tyurin & R (NeurIPS '23)	✓	✓	0	✓		**Inf**	✗	✓
Accelerated Rennala SGD Tyurin & R (NeurIPS '23)	✗	✓	0		✓	**Inf**	✗	✓
Shadowheart SGD Tyurin, Pozzi, Ilin & R '24	✗	✓	✓	✓		**Inf**	✗	✓
Freya PAGE Tyurin, Gruntkowska & R '24	✗	✓	0	✓		**Finite**	✗	✓ big data regime
Freya SGD Tyurin, Gruntkowska & R '24	✗	✓	0	✓		**Finite**	✗	✗
Fragile SGD Tyurin & R '24	✗	✓	✓	✓		**Inf**	✓	nearly
Amelie SGD Tyurin & R '24	✓	✓	✓	✓		**Inf**	✓	✓

Rennala SGD

Algorithmic idea: Minibatch SGD with asynchronous minibatch collection



Upper Bound

Theorem (informal)

Assume data homogeneity and zero communication times.
Then Rennala SGD solves the problem in

$$96 \times \min_{m\in\{1,\dots,n\}} \left(\frac{1}{m}\sum_{i=1}^{m}\frac{1}{\tau_i}\right)^{-1}\left(\frac{L\Delta}{\varepsilon}+\frac{L\Delta\sigma^2}{\varepsilon^2 m}\right)$$

seconds.

Gradient of f is L-Lipschitz

$\Delta := f(x^0) - \inf f$

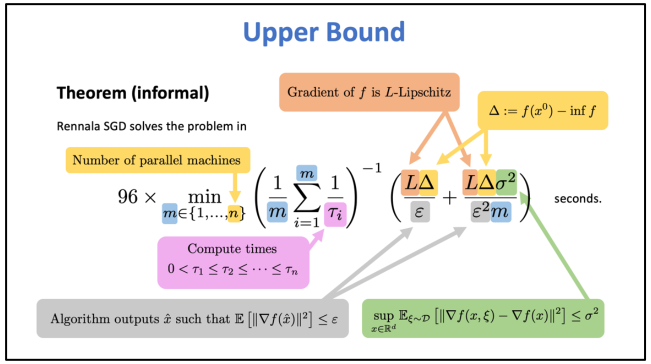
Number of parallel machines

Compute times $0 < \tau_1 \leq \tau_2 \leq \dots \leq \tau_n$

Algorithm outputs $\hat{x}$ such that $\mathbb{E}\left[\|\nabla f(\hat{x})\|^2\right] \leq \varepsilon$

$\sup_{x\in\mathbb{R}^d} \mathbb{E}_{\xi\sim\mathcal{D}}\left[\|\nabla f(x,\xi) - \nabla f(x)\|^2\right] \leq \sigma^2$

Matching Lower Bound



Theorem (informal)

It is not possible to design a method that will find a solution faster than in

$$\Omega\left(\min_{m \in \{1,\dots,n\}} \left(\frac{1}{m}\sum_{i=1}^{m}\frac{1}{\tau_i}\right)^{-1}\left(\frac{L\Delta}{\varepsilon} + \frac{L\Delta\sigma^2}{\varepsilon^2 m}\right)\right)$$

seconds.

Rennala SGD = first optimal parallel SGD

Classical Oracle: Keeps Track of # Iterations

Function class

Distribution governing noise

Oracle clas

Algorithm class

Protocol 1 Classical Ora le Protocol

1: **Input:** function $f \in \mathcal{F}$ oracle an

2: **for** $k = 0, \ldots, \infty$ **do**

3: $x^k = A^k(g^1, \ldots$ $\triangleright\ x^0 = A^0$ for $k = 0$.

4: $g^{k+1} = O($

5: **end for**

stochastic gradient: $g^{k+1} = \nabla f(x^k, \xi^{k+1})$

Natural for sequential methods, where a single worker does all the work!

Iteration com measure):

$$\mathfrak{m}_{\text{oracle}}(\mathcal{A}, \mathcal{F}) \quad \sup_{\in \mathcal{A}} \ \sup_{f \in \mathcal{F}} \ \sup_{(O,\mathcal{D}) \in \mathcal{O}(f)} \inf\left\{k \in \mathbb{N} \,\middle|\, \mathbb{E}\left[\|\nabla f(x^k)\|^2\right] \leq \varepsilon\right\}$$

[Nemirovsky and Yudin, 1983] [Nesterov, 2018]
[Carmon et al, 2020] [Arjevani et al, 2022]

New Oracle: Keeps Track of Time

Protocol 2 Time Oracle Protocol

1: **Input:** functions $f \in \mathcal{F}$, oracle and distribution $(O, \mathcal{D}) \in \mathcal{O}($ … $\in \mathcal{A}$
2: $s^0 = 0$
3: **for** $k = 0, \dots, \infty$ **do**
4: $(t^{k+1}, x^k) = A^k(g^1, \dots, g^k),$ ▷ $t^{k+1} \geq t^k$
5: $(s^{k+1}, g^{k+1}) = O(t^{k+1}, x^k, s^k,$ …
6: **end for**

Iteration complex…

$$\mathfrak{m}_{\text{oracle}}(\mathcal{A}, \dots \quad \dots (f) \quad \dots \left\{k \in \mathbb{N} \,\middle|\, \mathbb{E}\left[\|\nabla f(x^k)\|^2\right] \leq \varepsilon\right\}$$

Time complexity … xity measure):

$$S_t := \left\{k \in \mathbb{N} \cup \{0\} \,\middle|\, t^k \leq t\right\}$$

$$\mathfrak{m}_{\text{time}}(\mathcal{A}, \mathcal{F}) := \inf_{A \in \mathcal{A}} \sup_{f \in \mathcal{F}} \sup_{(O,\mathcal{D}) \in \mathcal{O}(f)} \inf \left\{t \geq 0 \,\middle|\, \mathbb{E}\left[\inf_{k \in S_t} \|\nabla f(x^k)\|^2\right] \leq \varepsilon\right\}$$

Natural for parallel methods!

Data Homogeneous Regime

Method	Time Complexity
Minibatch SGD	$\tau_n \left(\frac{L\Delta}{\varepsilon} + \frac{\sigma^2 L\Delta}{n\varepsilon^2} \right)$
Asynchronous SGD (Cohen et al., 2021) (Koloskova et al., 2022) (Mishchenko et al., 2022)	$\left(\frac{1}{n} \sum_{i=1}^{n} \frac{1}{\tau_i} \right)^{-1} \left(\frac{L\Delta}{\varepsilon} + \frac{\sigma^2 L\Delta}{n\varepsilon^2} \right)$
Rennala SGD (Theorem 7.5)	$\min_{m\in[n]} \left[\left(\frac{1}{m} \sum_{i=1}^{m} \frac{1}{\tau_i} \right)^{-1} \left(\frac{L\Delta}{\varepsilon} + \frac{\sigma^2 L\Delta}{m\varepsilon^2} \right) \right]$
Lower Bound (Theorem 6.4)	$\min_{m\in[n]} \left[\left(\frac{1}{m} \sum_{i=1}^{m} \frac{1}{\tau_i} \right)^{-1} \left(\frac{L\Delta}{\varepsilon} + \frac{\sigma^2 L\Delta}{m\varepsilon^2} \right) \right]$

Experimental Results (Sample)

$$\tau_i = \sqrt{i} \text{ seconds}$$

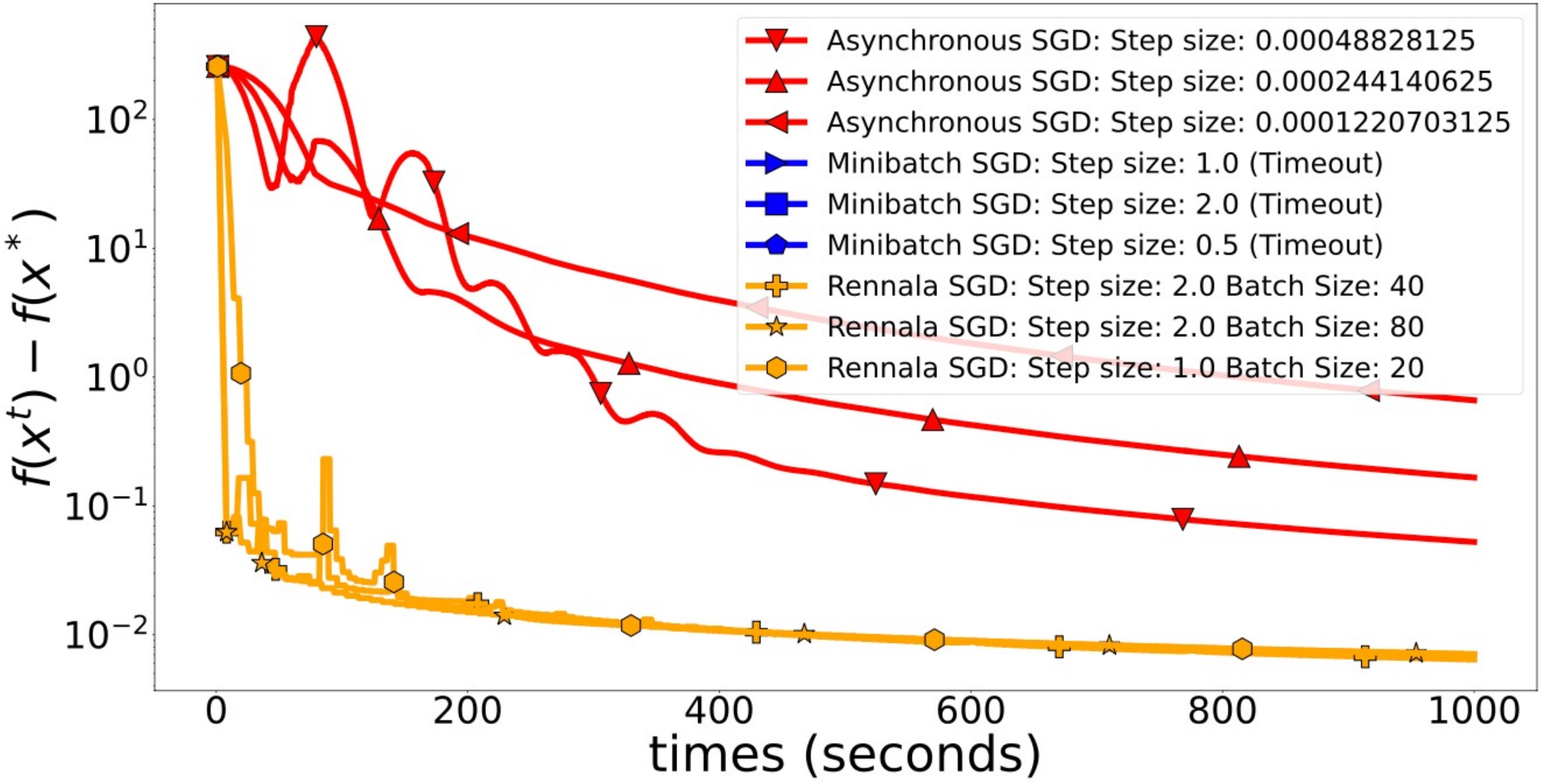
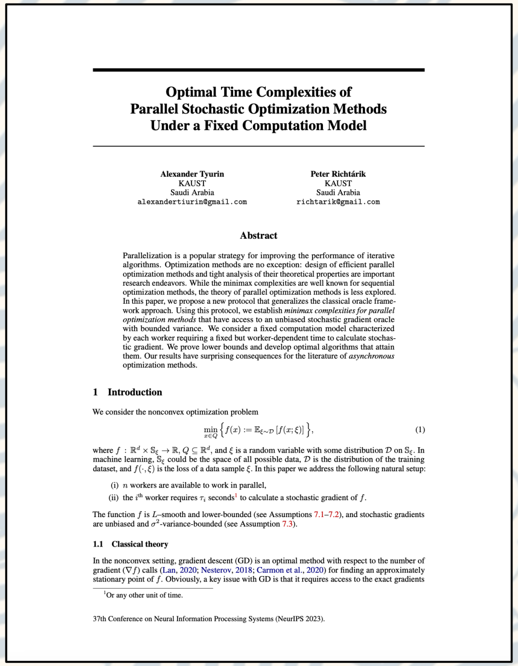


Figure 3: # of workers $n = 10000$.



Optimal Time Complexities of Parallel Stochastic Optimization Methods Under a Fixed Computation Model

Alexander Tyurin
KAUST
Saudi Arabia
alexandertiurin@gmail.com

Peter Richtárik
KAUST
Saudi Arabia
richtarik@gmail.com

Abstract

Parallelization is a popular strategy for improving the performance of iterative algorithms. Optimization methods are no exception: design of efficient parallel optimization methods and tight analysis of their theoretical properties are important research endeavors. While the minimax complexities are well known for sequential optimization methods, the theory of parallel optimization methods is less explored. In this paper, we propose a new protocol that generalizes the classical oracle framework approach. Using this protocol, we establish *minimax complexities for parallel optimization methods* that have access to an unbiased stochastic gradient oracle with bounded variance. We consider a fixed computation model characterized by each worker requiring a fixed but worker-dependent time to calculate stochastic gradient. We prove lower bounds and develop optimal algorithms that attain them. Our results have surprising consequences for the literature of *asynchronous* optimization methods.

1 Introduction

We consider the nonconvex optimization problem

$$\min_{x \in Q} \left\{ f(x) := \mathbb{E}_{\xi \sim \mathcal{D}} \left[f(x;\xi) \right] \right\}, \qquad (1)$$

where $f : \mathbb{R}^d \times \mathbb{S}_\xi \to \mathbb{R}$, $Q \subseteq \mathbb{R}^d$, and ξ is a random variable with some distribution $\mathcal{D}$ on $\mathbb{S}_\xi$. In machine learning, $\mathbb{S}_\xi$ could be the space of all possible data, $\mathcal{D}$ is the distribution of the training dataset, and $f(\cdot,\xi)$ is the loss of a data sample ξ. In this paper we address the following natural setup:

(i) n workers are available to work in parallel,

(ii) the i^{th} worker requires τ_i seconds[1] to calculate a stochastic gradient of f.

The function f is L–smooth and lower-bounded (see Assumptions 7.1–7.2), and stochastic gradients are unbiased and σ^2-variance-bounded (see Assumption 7.3).

1.1 Classical theory

In the nonconvex setting, gradient descent (GD) is an optimal method with respect to the number of gradient (∇f) calls (Lan, 2020; Nesterov, 2018; Carmon et al., 2020) for finding an approximately stationary point of f. Obviously, a key issue with GD is that it requires access to the exact gradients

[1]Or any other unit of time.

37th Conference on Neural Information Processing Systems (NeurIPS 2023).

Part 5
Two Extensions

Alexander Tyurin and P.R.

Optimal time complexities of parallel stochastic optimization methods under a fixed computation model

NeurIPS 2023

Extension 1
Handling Data Heterogeneity
(Malenia SGD)

Malenia SGD: Setup

$$\min_{x\in\mathbb{R}^d}\left\{f(x)=\frac{1}{n}\sum_{i=1}^{n} f_i(x)\right\}$$

$$f_i(x) := \mathbb{E}_{\xi\sim\mathcal{D}_i}\left[f_i(x,\xi)\right]$$

Optimal Parallel Stochastic Gradient Methods

	Data Heterogeneity ($\mathcal{D}_i$ different)	Compute Heterogeneity (τ_i different)	Communication Heterogeneity (θ_i different)	Smooth Nonconvex	Smooth Convex	Infinite / Finite Sum?	Supports Decentralized Setup?	Optimal Time Complexity?
Rennala SGD Tyurin & R (NeurIPS '23)	✗	✓	0	✓		Inf	✗	✓
Malenia SGD Tyurin & R (NeurIPS '23)	✓	✓	0	✓		Inf	✗	✓
Accelerated Rennala SGD Tyurin & R (NeurIPS '23)	✗	✓	0		✓	Inf	✗	✓
Shadowheart SGD Tyurin, Pozzi, Ilin & R '24	✗	✓	✓	✓		Inf	✗	✓
Freya PAGE Tyurin, Gruntkowska & R '24	✗	✓	0	✓		Finite	✗	✓ big data regime
Freya SGD Tyurin, Gruntkowska & R '24	✗	✓	0	✓		Finite	✗	✗
Fragile SGD Tyurin & R '24	✗	✓	✓	✓		Inf	✓	nearly
Amelie SGD Tyurin & R '24	✓	✓	✓	✓		Inf	✓	✓

The distributions $\mathcal{D}_1,\dots,\mathcal{D}_n$ are allowed to be different

Malenia SGD

Method 6 Malenia SGD

Input: starting point x^0, stepsize γ, parameter S
Run Method 7 in all workers
for $k = 0, 1, \ldots, K-1$ **do**
    Init $g_i^k = 0$ and $B_i = 0$
    while $\left(\frac{1}{n}\sum_{i=1}^{n}\frac{1}{B_i}\right)^{-1} < \frac{S}{n}$ **do**
        Wait for the next worker
        Receive gradient, iteration index, worker's index (g, k', i)
        if $k' = k$ **then**
            $g_i^k = g_i^k + g$
            $B_i = B_i + 1$
        end if
        Send (x^k, k) to the worker
    end while
    $g^k = \frac{1}{n}\sum_{i=1}^{n}\frac{1}{B_i}g_i^k$
    $x^{k+1} = x^k - \gamma g^k$
end for

Minibatch size

$$S = \max\left\{\left\lceil\frac{\sigma^2}{\varepsilon}\right\rceil, n\right\}$$

Method 7 Worker's Infinite Loop

Init $g = 0$, $k' = -1$, and worker's index i
while True **do**
    Send (g, k', i) to the server
    Receive (x^k, k) from the server
    $k' = k$
    $g = \widehat{\nabla} f_i(x^k; \xi), \quad \xi \sim \mathcal{D}$
end while

(Nonconvex) Data Heterogeneous Regime

Method	Time Complexity
Minibatch SGD	$\tau_n \left(\frac{L\Delta}{\varepsilon} + \frac{\sigma^2 L\Delta}{n\varepsilon^2} \right)$
Malenia SGD (Theorem A.4)	$\tau_n \frac{L\Delta}{\varepsilon} + \left(\frac{1}{n} \sum_{i=1}^{n} \tau_i \right) \frac{\sigma^2 L\Delta}{n\varepsilon^2}$
Lower Bound (Theorem A.2)	$\tau_n \frac{L\Delta}{\varepsilon} + \left(\frac{1}{n} \sum_{i=1}^{n} \tau_i \right) \frac{\sigma^2 L\Delta}{n\varepsilon^2}$

Extension 2
Handling the Convex Regime
(Accelerated Rennala SGD)

Accelerated Rennala SGD: Setup

Optimal Parallel Stochastic Gradient Methods

	Data Heterogeneity ($\mathcal{D}_i$ different)	Compute Heterogeneity (τ_i different)	Communication Heterogeneity (θ_i different)	Smooth Nonconvex	Smooth Convex	Infinite / Finite Sum?	Supports Decentralized Setup?	Optimal Time Complexity?
Rennala SGD Tyurin & R (NeurIPS '23)	✗	✓	0	✓		Inf	✗	✓
Malenia SGD Tyurin & R (NeurIPS '23)	✓	✓	0	✓		Inf	✗	✓
Accelerated Rennala SGD Tyurin & R (NeurIPS '23)	✗	✓	0		✓	Inf	✗	✓
Shadowheart SGD Tyurin, Pozzi, Ilin & R '24	✗	✓	✓	✓		Inf	✗	✓
Freya PAGE Tyurin, Gruntkowska & R '24	✗	✓	0	✓		Finite	✗	✓ big data regime
Freya SGD Tyurin, Gruntkowska & R '24	✗	✓	0	✓		Finite	✗	✗
Fragile SGD Tyurin & R '24	✗	✓	✓	✓		Inf	✓	nearly
Amelie SGD Tyurin & R '24	✓	✓	✓	✓		Inf	✓	✓

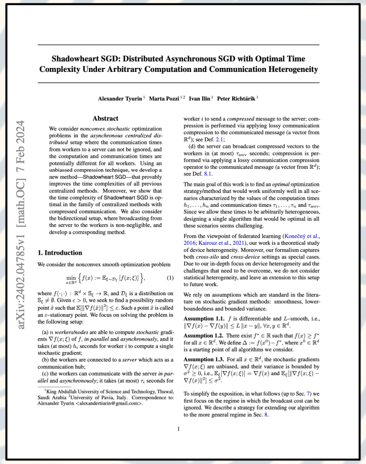
Convex (Data Homogeneous) Regime

Method	Time Complexity
Minibatch SGD	$\tau_n \left(\min\left\{\frac{\sqrt{L}R}{\sqrt{\varepsilon}}, \frac{M^2R^2}{\varepsilon^2}\right\} + \frac{\sigma^2R^2}{n\varepsilon^2}\right)$
Asynchronous SGD (Mishchenko et al., 2022)	$\left(\frac{1}{n}\sum_{i=1}^{n}\frac{1}{\tau_i}\right)^{-1}\left(\frac{LR^2}{\varepsilon} + \frac{\sigma^2R^2}{n\varepsilon^2}\right)$
(Accelerated) Rennala SGD (Theorems B.9 and B.11)	$\min\limits_{m\in[n]}\left[\left(\frac{1}{m}\sum_{i=1}^{m}\frac{1}{\tau_i}\right)^{-1}\left(\min\left\{\frac{\sqrt{L}R}{\sqrt{\varepsilon}}, \frac{M^2R^2}{\varepsilon^2}\right\} + \frac{\sigma^2R^2}{m\varepsilon^2}\right)\right]$
Lower Bound (Theorem B.4)	$\min\limits_{m\in[n]}\left[\left(\frac{1}{m}\sum_{i=1}^{m}\frac{1}{\tau_i}\right)^{-1}\left(\min\left\{\frac{\sqrt{L}R}{\sqrt{\varepsilon}}, \frac{M^2R^2}{\varepsilon^2}\right\} + \frac{\sigma^2R^2}{m\varepsilon^2}\right)\right]$
Lower Bound (Section M) (Woodworth et al., 2018)	$\tau_1 \min\left\{\frac{\sqrt{L}R}{\sqrt{\varepsilon}}, \frac{M^2R^2}{\varepsilon^2}\right\} + \left(\frac{1}{n}\sum_{i=1}^{n}\frac{1}{\tau_i}\right)^{-1}\frac{\sigma^2R^2}{n\varepsilon^2}$

∇f is L-Lipschitz, f is M-Lipschitz, and $\|x^0 - x^\star\| \leq R$

The End

Further Extensions



arXiv:2402.04785v1 [math.OC] 7 Feb 2024

Shadowheart SGD: Distributed Asynchronous SGD with Optimal Time Complexity Under Arbitrary Computation and Communication Heterogeneity

Alexander Tyurin [1] Marta Pozzi [1 2] Ivan Ilin [1] Peter Richtárik [1]

Abstract

We consider *nonconvex stochastic* optimization problems in the *asynchronous centralized distributed* setup where the communication times from workers to a server can not be ignored, and the computation and communication times are potentially different for all workers. Using an unbiassed compression technique, we develop a new method—Shadowheart SGD—that provably improves the time complexities of all previous centralized methods. Moreover, we show that the time complexity of Shadowheart SGD is optimal in the family of centralized methods with compressed communication. We also consider the bidirectional setup, where broadcasting from the server to the workers is non-negligible, and develop a corresponding method.

1. Introduction

We consider the nonconvex smooth optimization problem

$$\min_{x \in \mathbb{R}^d} \left\{ f(x) := \mathbb{E}_{\xi \sim \mathcal{D}_\xi} \left[f(x; \xi) \right] \right\}, \qquad (1)$$

where $f(\cdot;\cdot) : \mathbb{R}^d \times \mathbb{S}_\xi \to \mathbb{R}$, and $\mathcal{D}_\xi$ is a distribution on $\mathbb{S}_\xi \neq \emptyset$. Given $\varepsilon > 0$, we seek to find a possibility random point $\hat{x}$ such that $\mathbb{E}[\|\nabla f(\hat{x})\|^2] \leq \varepsilon$. Such a point $\hat{x}$ is called an ε–stationary point. We focus on solving the problem in the following setup:

(a) n *workers/nodes* are able to compute *stochastic* gradients $\nabla f(x;\xi)$ of f, *in parallel* and *asynchronously*, and it takes (at most) h_i seconds for worker i to compute a single stochastic gradient;

(b) the workers are connected to a *server* which acts as a communication hub;

(c) the workers can communicate with the server *in parallel* and *asynchronously*; it takes (at most) τ_i seconds for worker i to send a *compressed* message to the server; compression is performed via applying lossy communication compression to the communicated message (a vector from $\mathbb{R}^d$); see Def. 2.1;

(d) the server can broadcast compressed vectors to the workers in (at most) τ_{serv} seconds; compression is performed via applying a lossy communication compression operator to the communicated message (a vector from $\mathbb{R}^d$); see Def. 8.1.

The main goal of this work is to find an *optimal* optimization strategy/method that would work uniformly well in all scenarios characterized by the values of the computation times $h_1, \ldots, h_n$ and communication times $\tau_1, \ldots, \tau_n$ and τ_{serv}. Since we allow these times to be arbitrarily heterogeneous, designing a single algorithm that would be optimal in all these scenarios seems challenging.

From the viewpoint of federated learning (Konečný et al., 2016; Kairouz et al., 2021), our work is a theoretical study of device heterogeneity. Moreover, our formalism captures both *cross-silo* and *cross-device* settings as special cases. Due to our in-depth focus on device heterogeneity and the challenges that need to be overcome, we do not consider statistical heterogeneity, and leave an extension to this setup to future work.

We rely on assumptions which are standard in the literature on stochastic gradient methods: smoothness, lower-boundedness and bounded variance.

Assumption 1.1. f is differentiable and L–smooth, i.e., $\|\nabla f(x) - \nabla f(y)\| \leq L \|x - y\|, \forall x, y \in \mathbb{R}^d$.

Assumption 1.2. There exist $f^* \in \mathbb{R}$ such that $f(x) \geq f^*$ for all $x \in \mathbb{R}^d$. We define $\Delta := f(x^0) - f^*$, where $x^0 \in \mathbb{R}^d$ is a starting point of all algorithms we consider.

Assumption 1.3. For all $x \in \mathbb{R}^d$, the stochastic gradients $\nabla f(x;\xi)$ are unbiased, and their variance is bounded by $\sigma^2 \geq 0$, i.e., $\mathbb{E}_\xi[\nabla f(x;\xi)] = \nabla f(x)$ and $\mathbb{E}_\xi[\|\nabla f(x;\xi) - \nabla f(x)\|^2] \leq \sigma^2$.

To simplify the exposition, in what follows (up to Sec. 7) we first focus on the regime in which the broadcast cost can be ignored. We describe a strategy for extending our algorithm to the more general regime in Sec. 8.

[1] King Abdullah University of Science and Technology, Thuwal, Saudi Arabia [2] University of Pavia, Italy. Correspondence to: Alexander Tyurin <alexandertiurin@gmail.com>.

Shadowheart SGD

Optimal Parallel SGD
under Compute Heterogeneity
& Communication Heterogeneity

Shadowheart SGD: Setup

$$\min_{x \in \mathbb{R}^d} \left\{ f(x) = \frac{1}{n} \sum_{i=1}^{n} f_i(x) \right\}$$

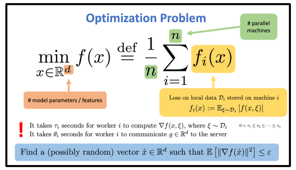
$$f_i(x) := \mathbb{E}_{\xi \sim \mathcal{D}_i} \left[f_i(x, \xi) \right]$$

Optimal Parallel Stochastic Gradient Methods

	Data Heterogeneity ($\mathcal{D}_i$ different)	Compute Heterogeneity (τ_i different)	Communication Heterogeneity (θ_i different)	Smooth Nonconvex	Smooth Convex	Infinite / Finite Sum?	Supports Decentralized Setup?	Optimal Time Complexity?
Rennala SGD Tyurin & R (NeurIPS '23)	✗	✓	0	✓		**Inf**	✗	✓
Malenia SGD Tyurin & R (NeurIPS '23)	✓	✓	0	✓		**Inf**	✗	✓
Accelerated Rennala SGD Tyurin & R (NeurIPS '23)	✗	✓	0		✓	**Inf**	✗	✓
Shadowheart SGD Tyurin, Pozzi, Ilin & R '24	✗	✓	✓	✓		**Inf**	✗	✓
Freya PAGE Tyurin, Gruntkowska & R '24	✗	✓	0	✓		**Finite**	✗	✓ big data regime
Freya SGD Tyurin, Gruntkowska & R '24	✗	✓	0	✓		**Finite**	✗	✗
Fragile SGD Tyurin & R '24	✗	✓	✓	✓		**Inf**	✓	nearly
Amelie SGD Tyurin & R '24	✓	✓	✓	✓		**Inf**	✓	✓

$\mathcal{D}_1 = \cdots = \mathcal{D}_n$

Communication costs $\theta_1, \ldots, \theta_n$ are nonzero (and possibly different)



Shadowheart SGD

Unbiased compressor:

$$\mathbb{E}\left[\mathcal{C}_{ij}(g)\right] = g \quad \& \quad \mathbb{E}\left[\|\mathcal{C}_{ij}(g) - g\|^2\right] \le \omega \|g\|^2 \quad \forall g \in \mathbb{R}^d$$

Aggregation weight associated with worker *i*

$$w_i = \left(\omega b_i + \omega \frac{\sigma^2}{\varepsilon} + m_i \frac{\sigma^2}{\varepsilon}\right)^{-1}$$

# of compressed batches sent by worker *i* to the server

$$m_i = \left\lfloor \frac{t^\star}{\theta_i} \right\rfloor$$

Batch size to compress by worker *i*

$$b_i = \left\lfloor \frac{t^\star}{\tau_i} \right\rfloor$$

$$x^{k+1} = x^k - \gamma \cdot \frac{\sum_{i=1}^{n} w_i \sum_{j=1}^{m_i} \mathcal{C}_{ij}\left(\sum_{l=1}^{b_i} \nabla f(x^k, \xi_{il}^k)\right)}{\sum_{i=1}^{n} w_i m_i b_i}$$

$$\gamma = \frac{1}{2L}$$

$$\text{Equilibrium time: } t^\star : \left(\omega, \frac{\sigma^2}{\varepsilon}, (\tau_i)_{i=1}^n, (\theta_i)_{i=1}^n\right) \mapsto \mathbb{R}_+$$

Table 1: **Time Complexities of Centralized Distributed Algorithms.** Assume that it takes at most h_i seconds to worker i to calculate a stochastic gradient and $\dot{\tau}_i$ seconds to send *one coordinate/float* to server. Abbreviations: L = smoothness constant, ε = error tolerance, $\Delta = f(x^0) - f^*$, n = # of workers, d = dimension of the problem. We take the Rand$K$ compressor with $K = 1$ (Def. C.1) (as an example) in QSGD and Shadowheart SGD. Due to Property 5.2, the choice $K = 1$ is optimal for Shadowheart SGD up to a constant factor.

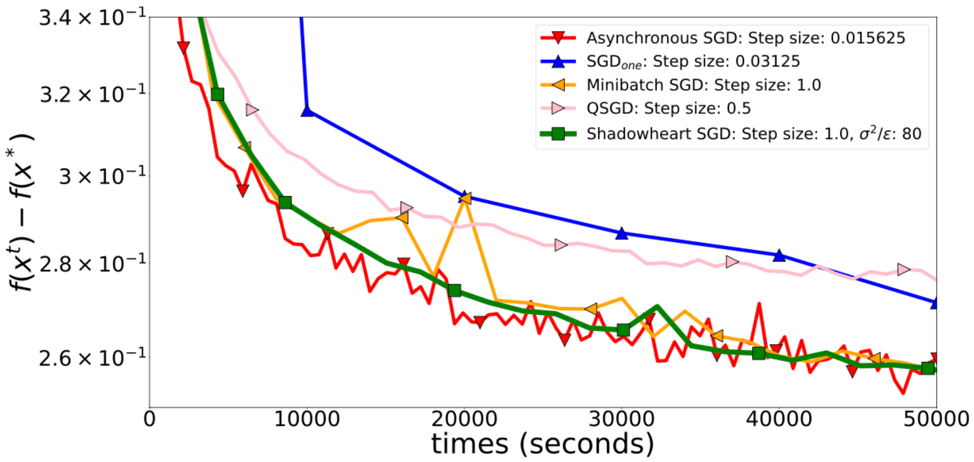
Method	Time Complexity	Time Complexities in Some Regimes: $\max\{h_n, \dot{\tau}_n\} \to \infty$, $\max\{h_i, \dot{\tau}_i\} < \infty \, \forall i < n$ (the last worker is slow)	Time Complexities in Some Regimes: $h_i = h, \dot{\tau}_i = \dot{\tau} \; \forall i \in [n]$ (equal performance)	Numerical Comparison[b] $\sigma^2/\varepsilon = 1$	10^3	10^6
Minibatch SGD (see (3))	$\max_{i \in [n]} \max\{h_i, d\dot{\tau}_i\} \left(\frac{L\Delta}{\varepsilon} + \frac{\sigma^2 L\Delta}{n\varepsilon^2}\right)$	∞ (non-robust)	$\max\{h, d\dot{\tau}, \frac{d\dot{\tau}\sigma^2}{n\varepsilon}, \frac{h\sigma^2}{n\varepsilon}\} \frac{L\Delta}{\varepsilon}$ (worse, e.g., when $\dot{\tau}$, d or n large)	$\times 10^3$	$\times 10^3$	$\times 10^4$
QSGD (see (7)) (Alistarh et al., 2017) (Khaled & Richtárik, 2020)	$\max_{i \in [n]} \max\{h_i, \dot{\tau}_i\} \left(\left(\frac{d}{n} + 1\right) \frac{L\Delta}{\varepsilon} + \frac{d\sigma^2 L\Delta}{n\varepsilon^2}\right)$	∞ (non-robust)	$\geq \frac{dh\sigma^2}{n\varepsilon} \frac{L\Delta}{\varepsilon}$ (worse, e.g., when ε small)	$\times 3$	$\times 10^2$	$\times 10^4$
Rennala SGD (Tyurin & Richtárik, 2023c), Asynchronous SGD (e.g., (Mishchenko et al., 2022))	$\geq \min_{j \in [n]} \max\left\{h_{\bar{\pi}_j}, d\dot{\tau}_{\bar{\pi}_j}, \frac{\sigma^2}{\varepsilon}\left(\sum_{i=1}^{j} \frac{1}{h_{\bar{\pi}_i}}\right)^{-1}\right\} \frac{L\Delta}{\varepsilon}$ [a]	$< \infty$ (robust)	$\geq \max\left\{h, d\dot{\tau}, \frac{h\sigma^2}{n\varepsilon}\right\} \frac{L\Delta}{\varepsilon}$ (worse, e.g., when $\dot{\tau}$, d or n large)	$\times 10^2$	$\times 10$	$\times 1.5$
Shadowheart SGD (see (9) and Alg. 1) (Corollary 4.4)	$t^*(d-1, \sigma^2/\varepsilon, [h_i, \dot{\tau}_i]_1^n) \frac{L\Delta}{\varepsilon}$ [c]	$< \infty$ (robust)	$\max\left\{h, \dot{\tau}, \frac{d\dot{\tau}}{n}, \sqrt{\frac{d\dot{\tau}h\sigma^2}{n\varepsilon}}, \frac{h\sigma^2}{n\varepsilon}\right\} \frac{L\Delta}{\varepsilon}$	$\times 1$	$\times 1$	$\times 1$

The time complexity of Shadowheart SGD is not worse than the time complexity of the competing centralized methods (see Sec. 6), and is *strictly* better in many regimes. We show that (12) is the *optimal time complexity* in the family of centralized methods with compression (see Sec. 7).

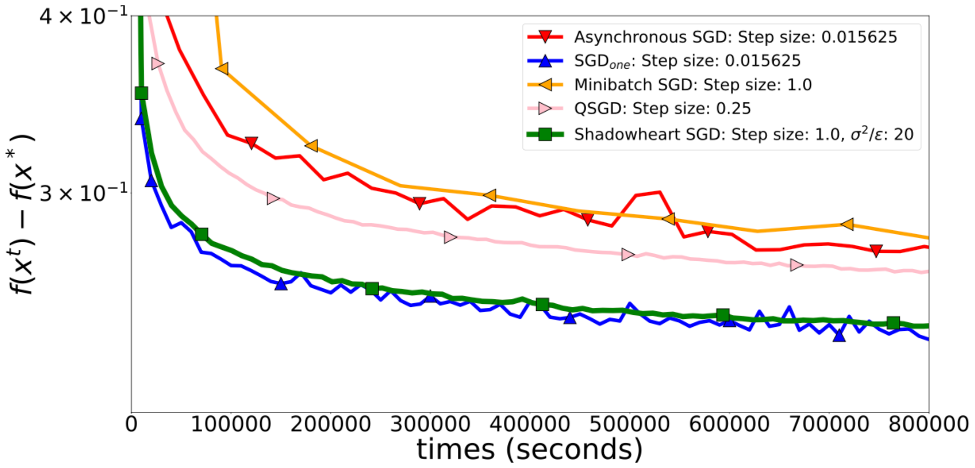
[a] Upper bound time complexities are not derived for Rennala SGD and Asynchronous SGD. However, we can derive the lower bound using Theorem N.5 with $\omega = 0$. One should take $d\dot{\tau}_i$ instead of τ_i when apply Theorem N.5 because these methods send d coordinates. $\bar{\pi}$ is a permutation that sorts $\max\{h_i, d\dot{\tau}_i\} : \max\{h_{\bar{\pi}_1}, d\dot{\tau}_{\bar{\pi}_1}\} \leq \cdots \leq \max\{h_{\bar{\pi}_n}, d\dot{\tau}_{\bar{\pi}_n}\}$

[b] We numerically compute time complexities for $d = 10^6$, $n = 10^3$, $h_i \sim U(0.1, 1)$, $\dot{\tau}_i \sim U(0.1, 1)$ (uniform i.i.d.), and three noise regimes $\sigma^2/\varepsilon \in \{1, 10^3, 10^6\}$. We report the factors by which the time complexities of the competing methods are worse compared to the time complexity of our method Shadowheart SGD. So, for example, Minibatch SGD, QSGD and Asynchronous SGD can be worse by the factors $\times 10^4$, $\times 10^4$, and $\times 10^2$, respectively.

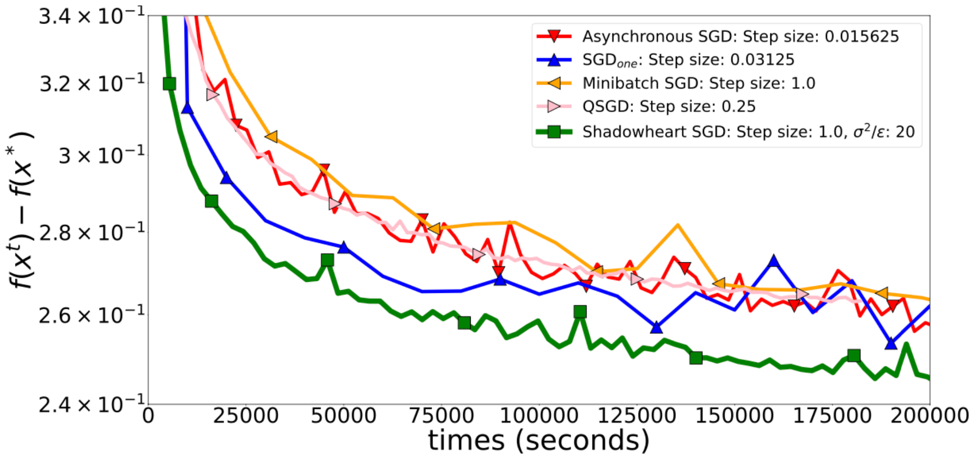
[c] The mapping t^* is defined in Def. 4.2.



Fast communication: $\dot{\theta}_i = \frac{\sqrt{i}}{d}$



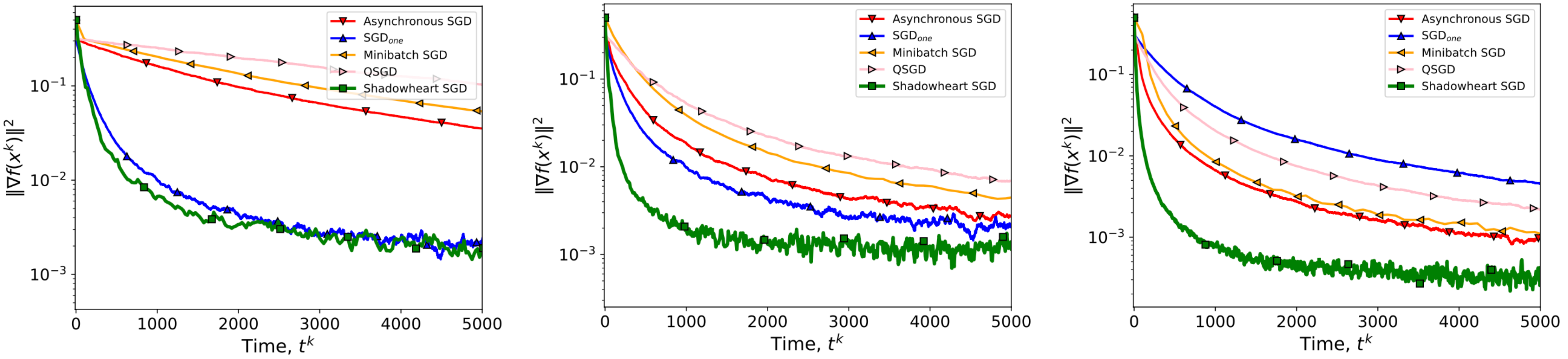
Slow communication: $\dot{\theta}_i = \frac{\sqrt{i}}{d^{1/2}}$



Medium-speed communication: $\dot{\theta}_i = \frac{\sqrt{i}}{d^{3/4}}$

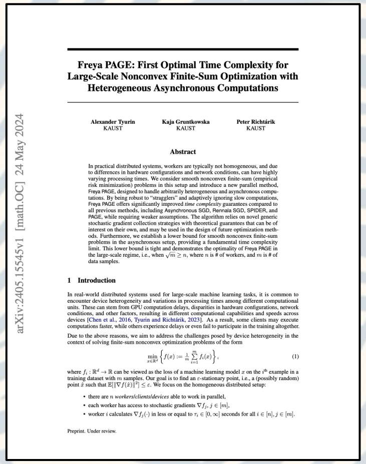
Computation times: $\tau_i = \sqrt{i}$ for all machines $i = 1, \ldots, n$

Shadowheart SGD: Adding More Workers…



(a) $n = 10$ (b) $n = 10^2$ (c) $n = 10^3$

$$\tau_i^k, \dot{\theta}_i^k \sim \mathrm{Uniform}(0.1, 1) \text{ for all } i \in \{1, \ldots, n\} \text{ and } k \geq 0$$



Freya PAGE: First Optimal Time Complexity for Large-Scale Nonconvex Finite-Sum Optimization with Heterogeneous Asynchronous Computations

Alexander Tyurin KAUST
Kaja Gruntkowska KAUST
Peter Richtárik KAUST

Abstract

In practical distributed systems, workers are typically not homogeneous, and due to differences in hardware configurations and network conditions, can have highly varying processing times. We consider smooth nonconvex finite-sum (empirical risk minimization) problems in this setup and introduce a new parallel method, Freya PAGE, designed to handle arbitrarily heterogeneous and asynchronous computations. By being robust to "stragglers" and adaptively ignoring slow computations, Freya PAGE offers significantly improved *time complexity* guarantees compared to all previous methods, including Asynchronous SGD, Rennala SGD, SPIDER, and PAGE, while requiring weaker assumptions. The algorithm relies on novel generic stochastic gradient collection strategies with theoretical guarantees that can be of interest on their own, and may be used in the design of future optimization methods. Furthermore, we establish a lower bound for smooth nonconvex finite-sum problems in the asynchronous setup, providing a fundamental time complexity limit. This lower bound is tight and demonstrates the optimality of Freya PAGE in the large-scale regime, i.e., when $\sqrt{m} \geq n$, where n is # of workers, and m is # of data samples.

1 Introduction

In real-world distributed systems used for large-scale machine learning tasks, it is common to encounter device heterogeneity and variations in processing times among different computational units. These can stem from GPU computation delays, disparities in hardware configurations, network conditions, and other factors, resulting in different computational capabilities and speeds across devices [Chen et al., 2016, Tyurin and Richtárik, 2023]. As a result, some clients may execute computations faster, while others experience delays or even fail to participate in the training altogether.

Due to the above reasons, we aim to address the challenges posed by device heterogeneity in the context of solving finite-sum nonconvex optimization problems of the form

$$\min_{x \in \mathbb{R}^d} \left\{ f(x) := \frac{1}{m} \sum_{i=1}^{m} f_i(x) \right\}, \quad (1)$$

where $f_i : \mathbb{R}^d \to \mathbb{R}$ can be viewed as the loss of a machine learning model x on the i^{th} example in a training dataset with m samples. Our goal is to find an ε-stationary point, i.e., a (possibly random) point $\hat{x}$ such that $\mathbb{E}[\|\nabla f(\hat{x})\|^2] \leq \varepsilon$. We focus on the homogeneous distributed setup:

- there are n *workers/clients/devices* able to work in parallel,
- each worker has access to stochastic gradients ∇f_j, $j \in [m]$,
- worker i calculates $\nabla f_j(\cdot)$ in less or equal to $\tau_i \in [0, \infty]$ seconds for all $i \in [n], j \in [m]$.

Preprint. Under review.

Shadowheart

Freya PAGE

Optimal Parallel SGD

for Large-Scale Finite-Sum Problems

Freya PAGE: Setup

$$\min_{x \in \mathbb{R}^d} \left\{ f(x) = \frac{1}{n} \sum_{i=1}^{n} f_i(x) \right\}$$

$$f_i(x) := \mathbb{E}_{\xi \sim \mathcal{D}_i} \left[f_i(x, \xi) \right]$$

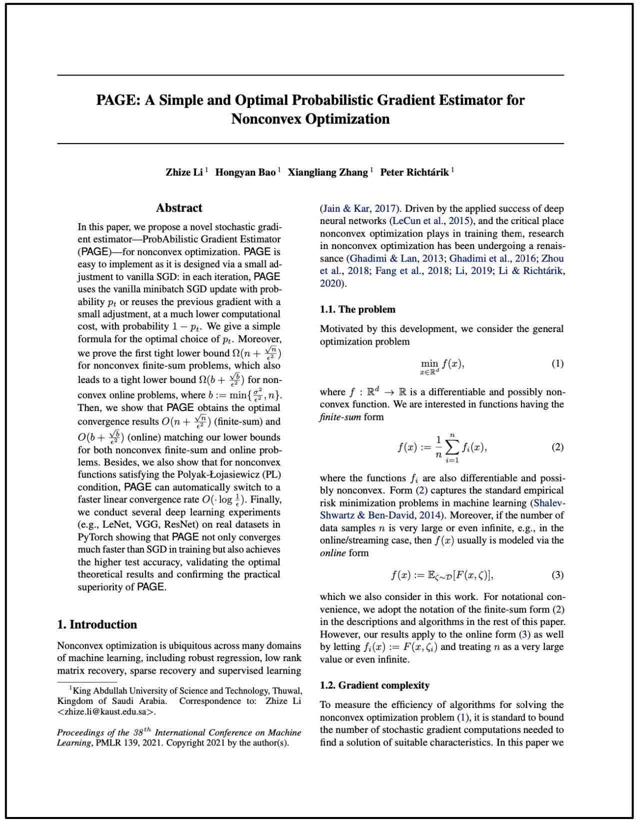
Optimal Parallel Stochastic Gradient Methods

	Data Heterogeneity ($\mathcal{D}_i$ different)	Compute Heterogeneity (τ_i different)	Communication Heterogeneity (θ_i different)	Smooth Nonconvex	Smooth Convex	Infinite / Finite Sum?	Supports Decentralized Setup?	Optimal Time Complexity?
Rennala SGD Tyurin & R (NeurIPS '23)	✗	✓	0	✓		Inf	✗	✓
Malenia SGD Tyurin & R (NeurIPS '23)	✓	✓	0	✓		Inf	✗	✓
Accelerated Rennala SGD Tyurin & R (NeurIPS '23)	✗	✓	0		✓	Inf	✗	✓
Shadowheart SGD Tyurin, Pozzi, Ilin & R '24	✗	✓	✓	✓		Inf	✗	✓
Freya PAGE Tyurin, Gruntkowska & R '24	✗	✓	0	✓		Finite	✗	✓ big data regime
Freya SGD Tyurin, Gruntkowska & R '24	✗	✓	0	✓		Finite	✗	✗
Fragile SGD Tyurin & R '24	✗	✓	✓	✓		Inf	✓	nearly
Amelie SGD Tyurin & R '24	✓	✓	✓	✓		Inf	✓	✓

$\mathcal{D}_1 = \cdots = \mathcal{D}_n$

$\mathcal{D}_i =$ uniform distribution over m outcomes

PAGE: Optimal Serial SGD for Finite-Sum Nonconvex Optimization



PAGE: A Simple and Optimal Probabilistic Gradient Estimator for Nonconvex Optimization

Zhize Li[1] Hongyan Bao[1] Xiangliang Zhang[1] Peter Richtárik[1]

Abstract

In this paper, we propose a novel stochastic gradient estimator—ProbAbilistic Gradient Estimator (PAGE)—for nonconvex optimization. PAGE is easy to implement as it is designed via a small adjustment to vanilla SGD: in each iteration, PAGE uses the vanilla minibatch SGD update with probability p_t or reuses the previous gradient with a small adjustment, at a much lower computational cost, with probability $1 - p_t$. We give a simple formula for the optimal choice of p_t. Moreover, we prove the first tight lower bound $\Omega(n + \frac{\sqrt{n}}{\epsilon^2})$ for nonconvex finite-sum problems, which also leads to a tight lower bound $\Omega(b + \frac{\sqrt{b}}{\epsilon^2})$ for nonconvex online problems, where $b := \min\{\frac{\sigma^2}{\epsilon^2}, n\}$. Then, we show that PAGE obtains the optimal convergence results $O(n + \frac{\sqrt{n}}{\epsilon^2})$ (finite-sum) and $O(b + \frac{\sqrt{b}}{\epsilon^2})$ (online) matching our lower bounds for both nonconvex finite-sum and online problems. Besides, we also show that for nonconvex functions satisfying the Polyak-Łojasiewicz (PL) condition, PAGE can automatically switch to a faster linear convergence rate $O(\cdot \log \frac{1}{\epsilon})$. Finally, we conduct several deep learning experiments (e.g., LeNet, VGG, ResNet) on real datasets in PyTorch showing that PAGE not only converges much faster than SGD in training but also achieves the higher test accuracy, validating the optimal theoretical results and confirming the practical superiority of PAGE.

1. Introduction

Nonconvex optimization is ubiquitous across many domains of machine learning, including robust regression, low rank matrix recovery, sparse recovery and supervised learning (Jain & Kar, 2017). Driven by the applied success of deep neural networks (LeCun et al., 2015), and the critical place nonconvex optimization plays in training them, research in nonconvex optimization has been undergoing a renaissance (Ghadimi & Lan, 2013; Ghadimi et al., 2016; Zhou et al., 2018; Fang et al., 2018; Li, 2019; Li & Richtárik, 2020).

1.1. The problem

Motivated by this development, we consider the general optimization problem

$$\min_{x \in \mathbb{R}^d} f(x), \qquad (1)$$

where $f : \mathbb{R}^d \to \mathbb{R}$ is a differentiable and possibly nonconvex function. We are interested in functions having the *finite-sum* form

$$f(x) := \frac{1}{n}\sum_{i=1}^{n} f_i(x), \qquad (2)$$

where the functions f_i are also differentiable and possibly nonconvex. Form (2) captures the standard empirical risk minimization problems in machine learning (Shalev-Shwartz & Ben-David, 2014). Moreover, if the number of data samples n is very large or even infinite, e.g., in the online/streaming case, then $f(x)$ usually is modeled via the *online* form

$$f(x) := \mathbb{E}_{\zeta \sim \mathcal{D}}[F(x, \zeta)], \qquad (3)$$

which we also consider in this work. For notational convenience, we adopt the notation of the finite-sum form (2) in the descriptions and algorithms in the rest of this paper. However, our results apply to the online form (3) as well by letting $f_i(x) := F(x, \zeta_i)$ and treating n as a very large value or even infinite.

1.2. Gradient complexity

To measure the efficiency of algorithms for solving the nonconvex optimization problem (1), it is standard to bound the number of stochastic gradient computations needed to find a solution of suitable characteristics. In this paper we

[1]King Abdullah University of Science and Technology, Thuwal, Kingdom of Saudi Arabia. Correspondence to: Zhize Li <zhize.li@kaust.edu.sa>.

Proceedings of the 38th International Conference on Machine Learning, PMLR 139, 2021. Copyright 2021 by the author(s).

$$\min_{x \in \mathbb{R}^d} \left\{ f(x) = \frac{1}{n}\sum_{i=1}^{n} f_i(x) \right\}$$

$$f_i(x) := \mathbb{E}_{\xi \sim \mathcal{D}_i}\left[f_i(x, \xi)\right]$$

$$\mathcal{D}_1 = \cdots = \mathcal{D}_n$$

$\mathcal{D}_i$ = uniform distribution over m outcomes

$$\min_{x \in \mathbb{R}^d} \left\{ f(x) = \frac{1}{m}\sum_{i=1}^{m} f_i(x) \right\}$$

(after butchering/redefining notation)

Zhize Li, Hongyan Bao, Xiangliang Zhang, and P.R.
PAGE: A simple and optimal probabilistic gradient estimator for nonconvex optimization
ICML 2021

Table 1: Comparison of the *worst-case time complexity* guarantees of methods that work with asynchronous computations in the setup from Section 1 (up to smoothness constants). We assume that $\tau_i \in [0, \infty]$ is the bound on the times required to calculate one stochastic gradient ∇f_j by worker i, $\tau_1 \leq \ldots \leq \tau_n$, and $m \geq n \log n$. Abbr: $\delta^0 := f(x^0) - f^*$, $m =$ # of data samples, $n =$ # of workers, $\varepsilon =$ error tolerance.

Method	**Worst-Case Time Complexity**	**Comment**
Hero GD (Soviet GD)	$\tau_1 m \frac{\delta^0}{\varepsilon} \quad \left(\tau_n \frac{m}{n} \frac{\delta^0}{\varepsilon}\right)$	Suboptimal
Hero PAGE (Soviet PAGE) [Li et al., 2021]	$\tau_1 m + \tau_1 \frac{\delta^0}{\varepsilon} \sqrt{m} \quad \left(\tau_n \frac{m}{n} + \tau_n \frac{\delta^0}{\varepsilon} \frac{\sqrt{m}}{n}\right)$	Suboptimal
SYNTHESIS [Liu et al., 2022]	—	Limitations: bounded gradient assumption, calculates the full gradients[(a)], suboptimal.[(b)]
Asynchronous SGD [Koloskova et al., 2022] [Mishchenko et al., 2022]	$\frac{\delta^0}{\varepsilon}\left(\left(\sum_{i=1}^{n} \frac{1}{\tau_i}\right)^{-1} \left(\frac{\sigma^2}{\varepsilon} + n\right)\right)$	Limitations: σ^2–bounded variance assumption, suboptimal when ε is small.
Rennala SGD [Tyurin and Richtárik, 2023]	$\frac{\delta^0}{\varepsilon} \min_{j \in [n]} \left(\left(\sum_{i=1}^{j} \frac{1}{\tau_i}\right)^{-1} \left(\frac{\sigma^2}{\varepsilon} + j\right)\right)$	Limitations: σ^2–bounded variance assumption, suboptimal when ε is small.
Freya PAGE (Theorems 7 and 8)	$\min_{j \in [n]} \left(\left(\sum_{i=1}^{j} \frac{1}{\tau_i}\right)^{-1} (m + j)\right) + \frac{\delta^0}{\varepsilon} \min_{j \in [n]} \left(\left(\sum_{i=1}^{j} \frac{1}{\tau_i}\right)^{-1} (\sqrt{m} + j)\right)$[(c)]	Optimal in the large-scale regime, i.e., $\sqrt{m} \geq n$ (see Section 5)
Lower bound (Theorem 10)	$\min_{j \in [n]} \left(\left(\sum_{i=1}^{j} \frac{1}{\tau_i}\right)^{-1} (m + j)\right) + \frac{\delta^0}{\sqrt{m}\varepsilon} \min_{j \in [n]} \left(\left(\sum_{i=1}^{j} \frac{1}{\tau_i}\right)^{-1} (m + j)\right)$	—

Freya PAGE has *universally* better guarantees than all previous methods: the dependence on ε is $\mathcal{O}(1/\varepsilon)$ (unlike Rennala SGD and Asynchronous SGD), the dependence on $\{\tau_i\}$ is harmonic-like and robust to slow workers (robust to $\tau_n \to \infty$) (unlike Soviet PAGE and SYNTHESIS), the assumptions are weak, and the time complexity of Freya PAGE is optimal when $\sqrt{m} \geq n$.

[(a)] In Line 3 of their Algorithm 3, they calculate the full gradient, assuming that it can be done for free and not explaining how.
[(b)] Their convergence rates in Theorems 1 and 3 depend on a bound on the delays Δ, which in turn depends on the performance of the slowest worker. Our method does not depend on the slowest worker if it is too slow (see Section 4.3), which is required for optimality.
[(c)] We prove better time complexity in Theorem 6, but this result requires the knowledge of $\{\tau_i\}$ in advance, unlike Theorems 7 and 8.

Algorithm 1 Freya PAGE

Parameters: starting point $x^0 \in \mathbb{R}^d$, learning rate $\gamma > 0$, minibatch size $S \in \mathbb{N}$, probability $p \in (0, 1]$, initialization $g^0 = \nabla f(x^0)$ using $\textsf{ComputeGradient}(x^0)$ (Alg. 2)
for $k = 0, 1, \ldots, K - 1$ **do**
  $x^{k+1} = x^k - \gamma g^k$
  Sample $c^k \sim \text{Bernoulli}(p)$
  if $c^k = 1$ **then** (with probability p)
    $\nabla f(x^{k+1}) = \textsf{ComputeGradient}(x^{k+1})$ (Alg. 2)
    $g^{k+1} = \nabla f(x^{k+1})$
  else (with probability $1 - p$)
    $\frac{1}{S} \sum_{i \in \mathcal{S}^k} \left(\nabla f_i(x^{k+1}) - \nabla f_i(x^k)\right) = \textsf{ComputeBatchDifference}(S, x^{k+1}, x^k)$ (Alg. 3)
    $g^{k+1} = g^k + \frac{1}{S} \sum_{i \in \mathcal{S}^k} \left(\nabla f_i(x^{k+1}) - \nabla f_i(x^k)\right)$
  end if
end for
(note): $\mathcal{S}^k$ is a set of i.i.d. indices that are sampled from $[m]$, *uniformly with replacement*, $\left|\mathcal{S}^k\right| = S$

Algorithm 2 ComputeGradient(x)

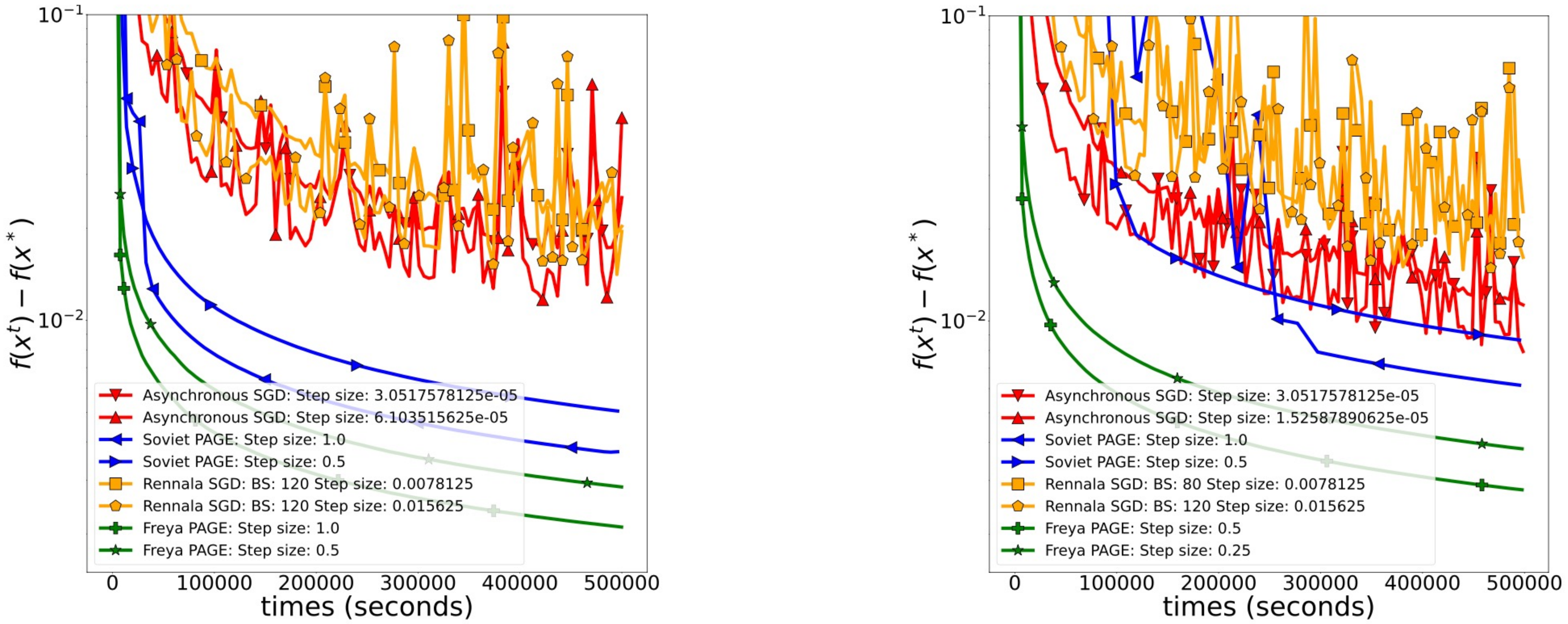
Input: point $x \in \mathbb{R}^d$
Init $g = 0 \in \mathbb{R}^d$, set $\mathcal{M} = \emptyset$
Broadcast x to all workers
For each worker $i \in [n]$, sample j from $[m]$ uniformly and ask it to calculate $\nabla f_j(x)$
while $\mathcal{M} \neq [m]$ **do**
    Wait for $\nabla f_p(x)$ from a worker
    if $p \in [m] \backslash \mathcal{M}$ **then**
        $g \leftarrow g + \frac{1}{m} \nabla f_p(x)$
        Update $\mathcal{M} \leftarrow \mathcal{M} \cup \{p\}$
    end if
    Sample j from $[m] \backslash \mathcal{M}$ uniformly and ask this worker to calculate $\nabla f_j(x)$
end while
Return $g = \frac{1}{m} \sum_{i=1}^{m} \nabla f_i(x)$

Algorithm 3 ComputeBatchDifference(S, x, y)

Input: batch size $S \in \mathbb{N}$, points $x, y \in \mathbb{R}^d$
Init $g = 0 \in \mathbb{R}^d$
Broadcast x, y to all workers
For each worker, sample j from $[m]$ uniformly and ask it to calculate $\nabla f_j(x) - \nabla f_j(y)$
for $i = 1, 2, \dots, S$ **do**
    Wait for $\nabla f_p(x) - \nabla f_p(y)$ from a worker
    $g \leftarrow g + \frac{1}{S}(\nabla f_p(x) - \nabla f_p(y))$
    Sample j from $[m]$ uniformly and ask this worker to calculate $\nabla f_j(x) - \nabla f_j(y)$
end for
Return g

Notes: i) the workers can aggregate ∇f_p locally, and the algorithm can call AllReduce once to collect all calculated gradients. ii) By splitting $[m]$ into blocks, instead of one ∇f_p, we can ask the workers to calculate the sum of one block in Alg. 2 (and use a similar idea in Alg. 3).

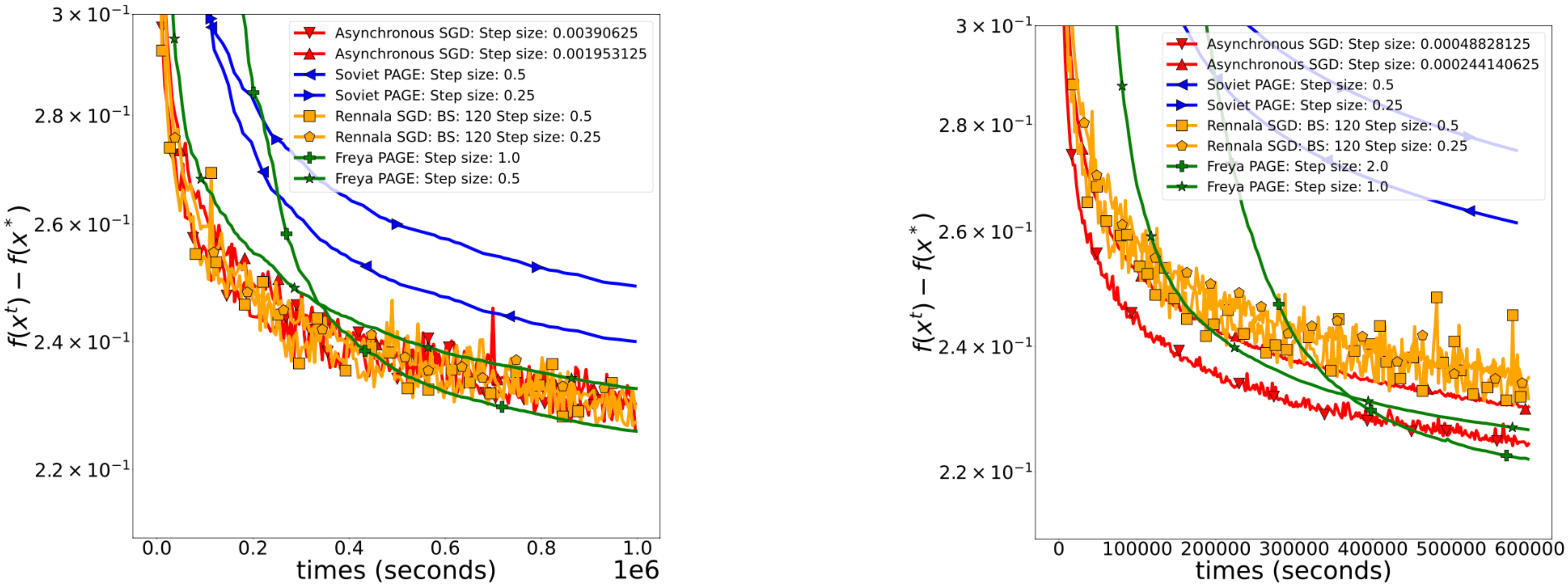
Freya PAGE: Experiment 1



(a) $n = 1000$ (b) $n = 10000$

Figure 1: Experiments with nonconvex quadratic optimization tasks. We plot function suboptimality against elapsed time.

Freya PAGE: Experiment 2



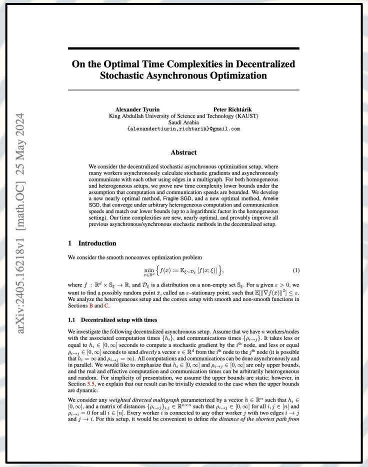
(a) $n = 100$ (b) $n = 10000$

Figure 2: Experiments with the logistic regression problem on the `MNIST` dataset.

Freya PAGE: Experiment 2

Table 2: Mean and variance of algorithm accuracies on the MNIST test set during the final 100K seconds of the experiments from Figure 2b.

Method	Accuracy	Variance of Accuracy
Asynchronous SGD [Koloskova et al., 2022] [Mishchenko et al., 2022]	92.60	5.85e-07
Soviet PAGE [Li et al., 2021]	92.31	1.62e-07
Rennala SGD [Tyurin and Richtárik, 2023]	92.37	3.12e-06
Freya PAGE	**92.66**	**1.01e-07**



arXiv:2405.16218v1 [math.OC] 25 May 2024

On the Optimal Time Complexities in Decentralized Stochastic Asynchronous Optimization

Alexander Tyurin **Peter Richtárik**
King Abdullah University of Science and Technology (KAUST)
Saudi Arabia
{alexandertiurin,richtarik}@gmail.com

Abstract

We consider the decentralized stochastic asynchronous optimization setup, where many workers asynchronously calculate stochastic gradients and asynchronously communicate with each other using edges in a multigraph. For both homogeneous and heterogeneous setups, we prove new time complexity lower bounds under the assumption that computation and communication speeds are bounded. We develop a new nearly optimal method, Fragile SGD, and a new optimal method, Amelie SGD, that converge under arbitrary heterogeneous computation and communication speeds and match our lower bounds (up to a logarithmic factor in the homogeneous setting). Our time complexities are new, nearly optimal, and provably improve all previous asynchronous/synchronous stochastic methods in the decentralized setup.

1 Introduction

We consider the smooth nonconvex optimization problem

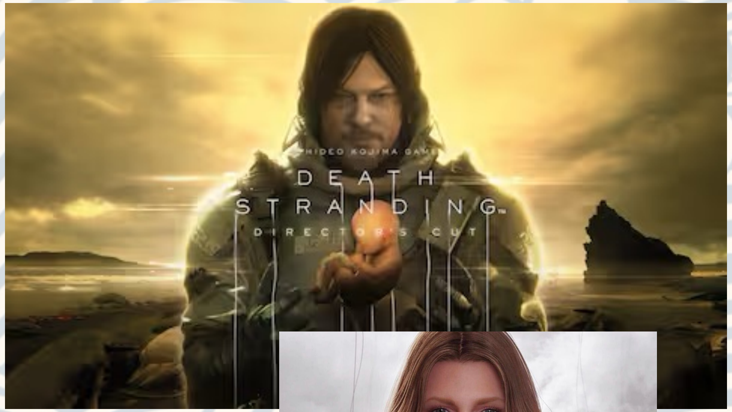
$$\min_{x \in \mathbb{R}^d} \Big\{ f(x) := \mathbb{E}_{\xi \sim \mathcal{D}_\xi} [f(x;\xi)] \Big\}, \quad (1)$$

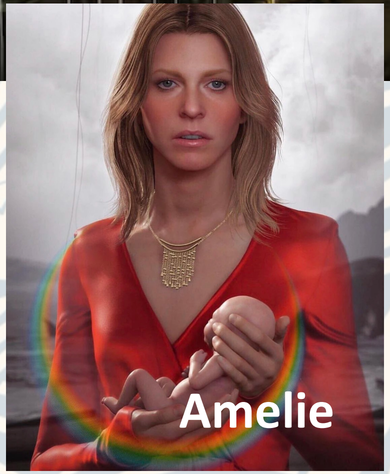
where $f : \mathbb{R}^d \times \mathbb{S}_\xi \to \mathbb{R}$, and $\mathcal{D}_\xi$ is a distribution on a non-empty set $\mathbb{S}_\xi$. For a given $\varepsilon > 0$, we want to find a possibly random point $\bar{x}$, called an ε–stationary point, such that $\mathbb{E}[\|\nabla f(\bar{x})\|^2] \le \varepsilon$. We analyze the heterogeneous setup and the convex setup with smooth and non-smooth functions in Sections B and C.

1.1 Decentralized setup with times

We investigate the following decentralized asynchronous setup. Assume that we have n workers/nodes with the associated computation times $\{h_i\}$, and communications times $\{\rho_{i\to j}\}$. It takes less or equal to $h_i \in [0,\infty]$ seconds to compute a stochastic gradient by the i^{th} node, and less or equal $\rho_{i\to j} \in [0,\infty]$ seconds to send *directly* a vector $v \in \mathbb{R}^d$ from the i^{th} node to the j^{th} node (it is possible that $h_i = \infty$ and $\rho_{i\to j} = \infty$). All computations and communications can be done asynchronously and in parallel. We would like to emphasize that $h_i \in [0,\infty]$ and $\rho_{i\to j} \in [0,\infty]$ are only upper bounds, and the real and effective computation and communication times can be arbitrarily heterogeneous and random. For simplicity of presentation, we assume the upper bounds are static; however, in Section 5.5, we explain that our result can be trivially extended to the case when the upper bounds are dynamic.

We consider any *weighted directed multigraph* parameterized by a vector $h \in \mathbb{R}^n$ such that $h_i \in [0,\infty]$, and a matrix of distances $\{\rho_{i\to j}\}_{i,j} \in \mathbb{R}^{n\times n}$ such that $\rho_{i\to j} \in [0,\infty]$ for all $i,j \in [n]$ and $\rho_{i\to i} = 0$ for all $i \in [n]$. Every worker i is connected to any other worker j with two edges $i \to j$ and $j \to i$. For this setup, it would be convenient to define *the distance of the shortest path from*





Amelie SGD

Optimal Decentralized SGD under Computation & Communication Heterogeneity

Decentralized Setup: Amelie SGD

Method	The Worst-Case Time Complexity Guarantees	Comment
Minibatch SGD	$\frac{L\Delta}{\varepsilon} \max\left\{\left(1+\frac{\sigma^2}{n\varepsilon}\right)\max\{\max_{i,j\in[n]}\tau_{i\to j}, \max_{i\in[n]} h_i\}\right\}$	suboptimal if σ^2/ε is large
RelaySGD, Gradient Tracking (Vogels et al., 2021) (Liu et al., 2024)	$\geq \frac{\max_{i\in[n]} L_i\Delta}{\varepsilon}\frac{\sigma^2}{n\varepsilon}\max_{i\in[n]} h_i$	requires local L_i-smooth. of f_i, suboptimal if σ^2/ε is large (even if $\max_{i\in[n]} L_i = L$)
Asynchronous SGD (Even et al., 2024)	—	requires similarity of the functions $\{f_i\}$, requires local L_i-smooth. of f_i
Amelie SGD and Lower Bound (Thm. 7 and Cor. 2)	$\frac{L\Delta}{\varepsilon}\max\left\{\max_{i,j\in[n]}\tau_{i\to j}, \max_{i\in[n]} h_i, \frac{\sigma^2}{n\varepsilon}\left(\frac{1}{n}\sum_{i=1}^{n} h_i\right)\right\}$	Optimal up to a constant factor

The End
(for real)