Delay-agnostic Asynchronous Coordinate Update Algorithm

Xuyang Wu [1] **Changxin Liu** [1] **Sindri Magnússon** [2] **Mikael Johansson** [1]

Abstract

We propose a delay-agnostic asynchronous coordinate update algorithm (DEGAS) for computing operator fixed points, with applications to asynchronous optimization. DEGAS includes novel asynchronous variants of ADMM and block-coordinate descent as special cases. We prove that DEGAS converges with both bounded and unbounded delays under delay-free parameter conditions. We also validate by theory and experiments that DEGAS adapts well to the actual delays. The effectiveness of DEGAS is demonstrated by numerical experiments on classification problems.

1. Introduction

Many popular algorithms in machine learning, optimization, and game theory can be formulated as fixed point iterations

$$\mathbf{x}(k+1) = \mathrm{T}(\mathbf{x}(k)), \tag{1}$$

where k is the iteration index, $\mathbf{x}(k) \in \mathbb{R}^d$ is the iterate at iteration k, and $\mathrm{T} : \mathbb{R}^d \to \mathbb{R}^d$ is an operator. For example, the gradient descent method for minimizing a differentiable function f is on the form (1) with $\mathrm{T}(\mathbf{x}) = \mathbf{x} - \gamma \nabla f(\mathbf{x})$ for some positive step-size parameter $\gamma > 0$.

In machine learning applications, the problem dimension is sometimes so large that evaluating the full operator T in each iteration is impractical. For these problems, coordinate update methods (Nesterov, 2012; Wright, 2015) have proven to be very competitive. These methods split the decision vector $\mathbf{x}$ into multiple blocks, $\mathbf{x} = (x_1, \ldots, x_m)$ and only update one block i in each iteration, *i.e.* they set

$$x_i(k+1) = \mathrm{T}_i(\mathbf{x}(k)), \tag{2}$$

while $x_j(k+1) = x_j(k)$ for all $j \neq i$. Here, T_i is the ith block of T such that $\mathrm{T}(\mathbf{x}) = (\mathrm{T}_1(\mathbf{x}), \ldots, \mathrm{T}_m(\mathbf{x}))$. In many cases, the cost of computing T_i can be much lower than that of computing the whole T (Nesterov, 2012).

A natural approach for accelerating coordinate update methods is to implement them on multiple processors/machines in a distributed environment. For example, in each iteration we may let every processor compute T_i for a randomly chosen block i, update all the selected blocks, and then go to the next iteration (Richtárik & Takáč, 2016). We consider this to be a synchronous update, since all the processors are synchronized and the algorithm does not proceed to the next iteration until all processors finish their work. Due to the use of multiple processors, synchronous coordinate update methods can converge significantly faster than the centralized coordinate update (2). However, their convergence speed is bottlenecked by the slowest processor and they are sensitive to single-node failures. In contrast, asynchronous coordinate update methods eliminate the need for global synchronization and can be more efficient and robust.

This paper focuses on asynchronous coordinate updates. Although these are useful in a wide range of sciences, this paper only discusses applications in optimization and ML.

1.1. Related work

In the past few decades, there has been a growing interest in developing parallel and asynchronous machine learning algorithms. As a part of this effort, a large number of asynchronous and distributed optimization algorithms with strong practical performance have been developed, including Async-SGD (Recht et al., 2011), Asynchronous ADMM (Zhang & Kwok, 2014), PIAG (Aytekin et al., 2016; Sun et al., 2019; Feyzmahdavian & Johansson, 2021), Async-BCD (Liu et al., 2014; Wu et al., 2022a), DAve-RPG (Mishchenko et al., 2018), DAve-QN (Soori et al., 2020), and ADSAGA (Glasgow & Wootters, 2022). Most of these algorithms are tailored to specific computing architectures such as master-worker (Li et al., 2013) or shared-memory (Bertsekas & Tsitsiklis, 1989), while algorithms such as AsySPA (Zhang & You, 2019), DFAL (Aybat et al., 2015), and the Asynchronous primal-dual algorithm (Wu et al., 2017) consider general communication topologies.

Two of the most influential frameworks for asynchronous coordinate update methods are due to (Bertsekas, 1983) and (Peng et al., 2016), respectively. In contrast to the related

[1]Division of Decision and Control Systems, EECS, KTH Royal Institute of Technology, Stockholm, Sweden [2]Department of Computer and System Science, Stockholm University, Stockholm, Sweden. Correspondence to: Xuyang Wu <xuyangw@kth.se>.

Proceedings of the 40^{th} International Conference on Machine Learning, Honolulu, Hawaii, USA. PMLR 202, 2023.

work cited above, which focuses on solving specific classes of optimization problems, they consider asynchronous coordinate updates for the more general problem of finding fixed points of operators. Specifically, Bertsekas (1983) proposes the following asynchronous implementation of (2):

$$x_i(k+1) = \mathrm{T}_i(\hat{\mathbf{x}}(k)), \tag{3}$$

where $\hat{\mathbf{x}}(k) = (\hat{x}_1(k), \dots, \hat{x}_n(k))$ with each $\hat{x}_j(k) = x_j(k - \tau_j(k))$ for some integer $\tau_j(k) \geq 0$. Here, $\tau_j(k)$ represents the information delay from node j, i.e. the difference between the current iteration index and the index of the iterate block used for computing T_i. However, this framework rarely applies to machine learning problems, since it is only guaranteed to converge if T is contractive in a block-maximum norm; see § 2.1.1. Even for gradient descent iterations on quadratic optimization problems, this condition only holds if the Hessian is diagonally dominant.

The ARock framwork of Peng et al. (2016) considers the modified coordinate updates

$$x_i(k+1) = x_i(k) + \gamma(k)(\mathrm{T}_i(\hat{\mathbf{x}}(k)) - \hat{x}_i(k)), \tag{4}$$

where $\gamma(k) > 0$ is the step-size. Unlike (Bertsekas, 1983), ARock only requires T to be non-expansive and applies to modern algorithms like BCD (Nesterov, 2012) and ADMM (Boyd et al., 2011). However, like most asynchronous algorithms that use fixed step-sizes, such as PIAG (Aytekin et al., 2016) and Async-BCD (Liu et al., 2014), existing convergence results require that delays are uniformly bounded and rely on step-size restrictions for $\gamma(k)$ that depend on this (typically unknown) delay bound. This causes difficulties in practice: using a large delay bound (to ensure that it is valid) leads to a small step-size and an unnecessarily slow convergence. In addition, guarding against the maximum delay leads to overly conservative results if most delays are smaller than the maximum delay. Indeed, a number of recent papers report delay measurements for asynchronous optimization algorithms that show that real-world delays tend to be distributed in this way; see, *e.g.*, (Mishchenko et al., 2022; Wu et al., 2022a; Koloskova et al., 2022) and our own measurements in Figure 5 in Appendix K. As a specific example, Mishchenko et al. (2022) implement an asynchronous SGD on a 40-core CPU and report maximal and average delays of around 1200 and 40, respectively.

1.2. Contribution

In this paper, we propose an alternative way to perform asynchronous coordinate updates. This approach, which we call the DElay-aGnostic ASynchronous coordinate update (DEGAS) algorithm, adapts the updates (3) to a master-worker architecture (Li et al., 2013) and samples the update block uniformly at random. We show that with these modifications, the new algorithm preserves the advantages of (3) and (4) and avoids their drawbacks in the sense that

i) Like (3), DEGAS is free from parameters that depend on the delay. In this way, it avoids ARock's issues with hard-to-determine and conservative step-sizes. Moreover, by characterizing how the convergence of DEGAS is affected by the distribution of delays in a stochastic delay model, we show that convergence is faster when small delays are more likely than large delays. This is in contrast to ARock, whose convergence rate is dominated by the worst-case (largest) delay and whose performance does not improve even if the actual delays are much smaller. This can be observed by scrutinising the convergence bounds in (Peng et al., 2016) and is confirmed in our numerical results.

ii) DEGAS converges under the same conditions on T as ARock, and can therefore be used for parallel and asynchronous implementations of a wide range of modern optimization methods, including BCD and ADMM. We prove that DEGAS converges under both bounded and unbounded delays. For bounded delays, we provide an explicit convergence rate and show that the iterates of DEGAS converge faster than the best-known bound for ARock. To derive this result, we prove a linear rate for a general class of asynchronous sequences which significantly sharpens a lemma from (Feyzmahdavian & Johansson, 2021).

We illustrate the superior performance of DEGAS (including ADMM and BCD) in training of large scale models.

Notation and Preliminaries

We let $\mathbb{N}$ be the set of natural numbers, and $\mathbb{N}_0 = \mathbb{N} \cup \{0\}$. We denote $[m] = \{1, \dots, m\}$ for any $m \in \mathbb{N}$ and define the proximal operator of a function $R : \mathbb{R}^d \to \mathbb{R} \cup \{+\infty\}$ as $\mathrm{prox}_R(x) = \mathrm{argmin}_{y \in \mathbb{R}^d}\ R(y) + \frac{1}{2}\|y - x\|^2$. We call a differentiable function $f : \mathbb{R}^d \to \mathbb{R}$ L-smooth if $\|\nabla f(\mathbf{x}) - \nabla f(\mathbf{y})\| \leq L\|\mathbf{x} - \mathbf{y}\|\ \forall \mathbf{x}, \mathbf{y} \in \mathbb{R}^d$, and μ-strongly convex if $\langle \nabla f(\mathbf{x}) - \nabla f(\mathbf{y}), \mathbf{x} - \mathbf{y} \rangle \geq \mu\|\mathbf{x} - \mathbf{y}\|^2\ \forall \mathbf{x}, \mathbf{y} \in \mathbb{R}^d$. We use Id to denote the identity operator of proper dimension. For any operator $\mathrm{T} : \mathbb{R}^d \to \mathbb{R}^d$, $\mathrm{Fix}\,\mathrm{T} = \{\mathbf{x} : \mathbf{x} = \mathrm{T}\,\mathbf{x}\}$ represents its set of fixed-points. We use $\|\cdot\|$ to represent the Euclidean norm for vectors and the spectral norm for matrices. For any vector $\mathbf{x} = (x_1, \dots, x_m) \in \mathbb{R}^N$ and $w = (w_1, \dots, w_m) \in \mathbb{R}^m$ where each $x_i \in \mathbb{R}^{d_i}$ and $w_i > 0$, we define $\|\mathbf{x}\|_{b,\infty}^w = \max_{i \in [m]} \frac{\|x_i\|_i}{w_i}$ as the block-maximum norm, where each $\|\cdot\|_i$ can be any vector norm.

2. Algorithm and Main Result

In this section, we present our algorithm for finding operator fixed points, analyze its convergence, and highlight its advantages over ARock (Peng et al., 2016).

2.1. Algorithm

We adapt the asynchronous update (3) to the widely-used master-worker architecture (Li et al., 2013) for distributed

Algorithm 1 DEGAS

Setup: initial iterate $\mathbf{x}(0)$.
Initialization: the master sets $\mathbf{x} = \mathbf{x}(0)$ and broadcasts $\mathbf{x}$ to all workers.
while *not interrupted by master*: each worker $w \in [n]$ *asynchronously* and *continuously* **do**
  receive $\mathbf{x}$ from the master and assign $\mathbf{x}^w = \mathbf{x}$.
  sample $i \in [m]$ uniformly at random.
  compute $\mathrm{T}_i(\mathbf{x}^w)$.
  send $(\mathrm{T}_i(\mathbf{x}^w), i)$ to the master.
end while
while *not converged*: the master **do**
  receive $(\mathrm{T}_i(\mathbf{x}^w), i)$ from a worker w.
  update $x_i \leftarrow \mathrm{T}_i(\mathbf{x}^w)$.
  send $\mathbf{x}$ to the worker w.
end while

learning. Here, a master node stores the current model $\mathbf{x}$ and coordinates the work of n compute nodes. Each worker $w \in [n]$ asynchronously and continuously receives $\mathbf{x}$ from the master, stores it in the local variable $\mathbf{x}^w$, computes $\mathrm{T}_i(\mathbf{x}^w)$ for some $i \in [m]$ drawn uniformly at random, and returns $\mathrm{T}_i(\mathbf{x}^w)$ to the master. Once the master receives $\mathrm{T}_i(\mathbf{x}^w)$ from some worker w, it updates

$$x_i = \mathrm{T}_i(\mathbf{x}^w) \tag{5}$$

and pushes the updated model back to any idle workers. A detailed implementation is given in Algorithm 1, which we refer to as the DElay-aGnostic ASynchronous coordinate update (DEGAS) algorithm.

For the convenience of further discussion, we index the iterates by $k \in \mathbb{N}_0$, which represents the number of updates by the master, and use $i(k)$ to denote the updated block at time k. Note that in DEGAS, each $\mathbf{x}^w$ in (5) is a delayed iterate and equals to $\mathbf{x}(k - \tau(k))$ for some integer $\tau(k) \in [0, k]$. We refer to $\tau(k)$ as the delay at time k. In this way, the update at time $k \in \mathbb{N}_0$ can be equivalently rewritten as

$$x_j(k+1) = \begin{cases} \mathrm{T}_j(\mathbf{x}(k - \tau(k))), & j = i(k), \\ x_j(k), & \text{otherwise.} \end{cases} \tag{6}$$

2.1.1. Connection with existing works

DEGAS can be viewed as an adaption of the asynchronous update (3) to the master-worker architecture. They are logically equivalent except for the particular block selection rule in DEGAS. Existing work on (3) mainly focused on a setting where the number of blocks and processors (or workers) are identical and each processor updates a certain block. To guarantee convergence in this setting, it is common to require that T is contractive in the block-maximum norm (Bertsekas & Tsitsiklis, 1989), i.e. to satisfy

$$\| \mathrm{T}(\mathbf{x}) - \mathrm{T}(\mathbf{x}^\star) \|_{b,\infty}^w < c \| \mathbf{x} - \mathbf{x}^\star \|_{b,\infty}^w \tag{7}$$

for some $c \in (0, 1)$, some $\mathbf{x}^\star \in \operatorname{Fix} \mathrm{T}$ and for every $\mathbf{x} \in \mathbb{R}^d$, where the block-maximum norm $\| \cdot \|_{b,\infty}^w$ is defined in Section 1. The condition (7) is restrictive and only holds for very specific operators, e.g., (Frommer, 1991), (Bertsekas & Tsitsiklis, 1989), (Mehyar et al., 2007), (Moallemi & Van Roy, 2010), and (Hale et al., 2017). Even for the simple operator $\mathrm{T} = \mathrm{Id} - \frac{1}{L}\nabla f$ where $f(x) = \frac{1}{2}\mathbf{x}^T A \mathbf{x} + b^T \mathbf{x}$ for a symmetric and positive definite matrix $A \in \mathbb{R}^{m \times m}$ and a vector $b \in \mathbb{R}^m$ and $L = \|A\|_2$, the condition (7) is known to hold only when A is diagonally dominant. In the next subsection we will show that DEGAS, by letting each processor update a random block in (3), can converge under a much weaker condition.

The ARock framework (Peng et al., 2016) uses updates that are rather different. First, while (3) only depends on $\hat{\mathbf{x}}(k)$, the ARock updates (4) are based on both $\hat{\mathbf{x}}(k)$ and $x_i(k)$. To guarantee convergence, the maximally allowable step-size depends on the (usually unknown and large) worst-case delay, and decays quickly with the upper bound on the delays. This makes the algorithm difficult to tune and unnecessarily slow in practice. In contrast, DEGAS does not need access to the upper delay bound for tuning, but converges for all bounded delays. Moreover, Example 1 in (Feyzmahdavian et al., 2014) provides a comparison between two delayed gradient methods, which are special cases of DEGAS and ARock with one block and one worker, respectively. They show that for a simple problem, the method specialized from DEGAS strictly outperforms the one from ARock, which suggests the superiority of the algorithmic form of DEGAS. We admit that the ARock framework is more flexible because it allows for inconsistent read and write while DEGAS does not. Due to this reason, ARock can be implemented on both the master-worker and the shared memory system, while DEGAS can only be implemented in the former where inconsistent read and write can be practically avoided.

Some existing asynchronous optimization methods can also converge with step-sizes that do not rely on the worst-case delay. Their step-sizes can be categorized as 1) delay-free and fixed; 2) delay-free and diminishing; 3) delay-adaptive. We are only aware of four other asynchronous algorithms that converge with delay-free fixed step-sizes: the delayed proximal gradient method (Feyzmahdavian et al., 2014), the asynchronous ADMM (Zhang & Kwok, 2014), DAve-RPG (Mishchenko et al., 2018), and the asynchronous level bundle method (Iutzeler et al., 2020). The first three algorithms are different from DEGAS and, unfortunately, do not cover coordinate update methods like BCD and ADMM. Zhang & Kwok (2014) assume that all workers have the same probability of sending results to the master in each iteration, which is less practical. The works (Sra et al., 2016; Wu et al., 2022a; Cohen et al., 2021; Koloskova et al., 2022) avoid using the worst-case delay by adapting step-sizes to the actual delays or the errors caused by actual delays. Here,

Wu et al. (2022a) study PIAG and asynchronous BCD, while the others focus on asynchronous SGD. The works (Agarwal & Duchi, 2011; Zhou et al., 2018; Aviv et al., 2021) show convergence of the asynchronous SGD or its variants, under delay-free diminishing step-sizes that are effective in stochastic optimization but may lead to slow convergence when applied to deterministic optimization problems.

2.2. Convergence analysis

Throughout the paper, we assume that the actual delays are independent from the sequence of selected blocks.

Assumption 1. *The delay sequence $\{\tau(k)\}_{k\in\mathbb{N}_0}$ and the block sequence $\{i(k)\}_{k\in\mathbb{N}_0}$ are independent.*

Assumption 1 is common in many papers on asynchronous optimization, e.g., ARock (Peng et al., 2016), asynchronous SGD (Recht et al., 2011; Mishchenko et al., 2022), and asynchronous coordinate descent (Liu & Wright, 2015). However, as highlighted in Leblond et al. (2018), the assumption may not hold in practice if T_i is more expensive to compute for some block i than for the others. Possible approaches for relaxing Assumption 1 include before read labeling (Mania et al., 2017), after read labeling (Leblond et al., 2018), and single coordinate consistent ordering (Cheung et al., 2021).

We first consider the case where all delays are bounded.

Assumption 2 (Partial asynchrony)**.** *There exists some finite number $\bar{\tau}\in\mathbb{N}_0$ such that $\tau(k)\le\bar{\tau}$ for all $k\in\mathbb{N}_0$.*

We analyze two classes of operators T defined next.

Definition 1 (Averaged operator)**.** *The operator $\mathrm{T}:\mathbb{R}^d\to\mathbb{R}^d$ is an α-averaged operator if $\mathrm{T}=(1-\alpha)\,\mathrm{Id}+\alpha R$ for some $\alpha\in(0,1)$ and some non-expansive operator R.*

Definition 2 (Pseudo-contractive operator)**.** *The operator $\mathrm{T}:\mathbb{R}^d\to\mathbb{R}^d$ is pseudo-contractive with modulus $c\in(0,1)$ if $Fix\,\mathrm{T}\neq\emptyset$ and for any $\mathbf{x}^\star\in Fix\,\mathrm{T}$ and $\mathbf{x}\in\mathbb{R}^d$,*

$$\|\,\mathrm{T}(\mathbf{x})-\mathrm{T}(\mathbf{x}^\star)\|\le c\|\mathbf{x}-\mathbf{x}^\star\|. \tag{8}$$

Examples of averaged operators include the proximal operator prox_f of a closed and convex function f, the gradient descent operator $\mathrm{Id}-\gamma\nabla f$, $\gamma\in(0,2/L)$ of a convex and L-smooth f, the Douglas-Rachford splitting of two $1/2$-averaged operators, and the forward-backward splitting of a maximally monotone operator and a cocoercive operator. These operators may be pseudo-contractive under stronger conditions (Bauschke et al., 2011).

Theorem 1. *Let $\mathbf{x}^\star\in Fix\,\mathrm{T}$ and $\{\mathbf{x}(k)\}$ be generated by DEGAS under Assumptions 1–2. If T is averaged, then*

$$\min_{t\le k}\mathbb{E}\left[\|(\mathrm{Id}-\mathrm{T})(\mathbf{x}(t))\|\right]=O(1/k). \tag{9}$$

If T is pseudo-contractive with modulus $c\in(0,1)$ then

$$\mathbb{E}[\|\mathbf{x}(k)-\mathbf{x}^\star\|^2]\le\rho_a^k\|\mathbf{x}(0)-\mathbf{x}^\star\|^2 \tag{10}$$

holds for all $k\in\mathbb{N}_0$ where $\rho_a=(1-\frac{1-c^2}{m})^{\frac{1}{1+\bar{\tau}/m}}$.

Proof. See Appendix A. □

In Theorem 1, the expectation is taken over historical block selections. The linear rate (10) is derived by using Lemma 9 in Appendix A, which establishes a linear convergence rate for a class of asynchronous sequences that significantly sharpens the rate in (Feyzmahdavian & Johansson, 2021).

The rate in Theorem 1 is tight in the sense that it is of the same order as the best-known rates for the *centralized* coordinate update (2). When $\bar{\tau}=0$, the rate ρ_a in (10) reduces to the typical rate $\rho_c:=1-\frac{1-c^2}{m}$ of the centralized coordinate update. Moreover, for any $\epsilon>0$, to achieve $\mathbb{E}[\|\mathbf{x}(k)-\mathbf{x}^\star\|^2]\le\epsilon$, DEGAS requires at most

$$K_a(\epsilon):=(1+\bar{\tau}/m)K_c(\epsilon) \tag{11}$$

iterations, where $K_c(\epsilon)=\log_{1/\rho_c}\frac{\|\mathbf{x}(0)-\mathbf{x}^\star\|^2}{\epsilon}$ is the iteration complexity of the centralized coordinate update method for achieving the same accuracy.

Remark 1 (Linear speed-up)**.** *Suppose that $\bar{\tau}$ is proportional to the number of workers, n. This happens when workers cyclically return information to the master, and is a good approximation for many distributed architectures for small to moderate values of n. Then, by (11),*

$$K_a(\epsilon)=(1+\Theta(n)/m)K_c(\epsilon). \tag{12}$$

If the computation time of T_i dominates the per-iteration cost of DEGAS, then a single iteration of DEGAS takes $1/n$ of the time of a centralized coordinate update (2) (Peng et al., 2016). Combining this observation with (12) reveals that DEGAS needs

$$1/n+\Theta(1)/m \tag{13}$$

times that of the centralized coordinate update to achieve a given accuracy. Note that (13) is approximately inversely proportional to n when $m\gg n$. This linear speedup property (Peng et al., 2016) is a desirable feature of distributed optimization/learning algorithms.

Remark 2 (Comparison with ARock)**.** *Peng et al. (2016) establish a linear convergence rate $O(\rho^k)$ for ARock that is improved in (Feyzmahdavian & Johansson, 2021) to*

$$\rho=1-\frac{1-c^2}{m(1+6\left(\frac{\bar{\tau}}{m}+\sqrt{\frac{\bar{\tau}}{m}}\right))}\ge\rho_a^{\frac{1+\bar{\tau}/m}{1+6\left(\frac{\bar{\tau}}{m}+\sqrt{\frac{\bar{\tau}}{m}}\right)}}. \tag{14}$$

The inequality in (14) is established in Appendix B. Hence, to achieve the same accuracy, ARock needs at least $\frac{1+6(\bar{\tau}/m+\sqrt{\bar{\tau}/m})}{1+\bar{\tau}/m}$ times as many iterations of DEGAS. When $\bar{\tau}=m$, this is a factor of roughly 6.5.

2.2.1. SELF-ADAPTIVITY TO ACTUAL DELAYS

Since the maximal delay can be very large while most delays are significantly smaller (Mishchenko et al., 2022; Wu et al., 2022a; Koloskova et al., 2022), the ability to adapt to the actual delays and not be significantly slowed down by infrequent occurrences of larger delays is an attractive algorithm feature. We call this property *self-adaptivity* of an asynchronous algorithm to actual delays.

Unlike ARock, whose maximally allowable step-size decreases with the maximum delay, DEGAS includes no delay information in its parameters and intuitively has better self-adaptivity. However, in Theorem 1, the use of the worst-case delay in the analysis can give loose convergence rate bounds and does not indicate any advantage of a system in which the worst-case delay is rarely attained over one that tends to run with delays close to the worst-case all the time. To reveal how the actual delays rather than their upper bound affects the convergence of DEGAS, we consider delays described by the following stochastic model:

Assumption 3. *The delays $\{\tau(k)\}_{k\in\mathbb{N}_0}$ are i.i.d. with probability distribution $\mathcal{P}$, where*

$$\Pr(\tau(k)=i)=P_i,\ \forall i\in\{0,\dots,\bar{\tau}\},\tag{15}$$

with $\sum_{i=0}^{\bar{\tau}} P_i=1$ and $P_i\geq 0\ \forall i\in\{0,\dots,\bar{\tau}\}$.

As the next result shows, the convergence rate of Algorithm 1 under such delays can be characterized by

$$\phi(\rho)=\rho-\rho_c-\frac{c^2}{m}(\sum_{i=0}^{\bar{\tau}}P_i\rho^{-i}-1).$$

Theorem 2. *Suppose that Assumption 3 holds, $\bar{\tau}\geq 1$, and $P_0<1$. Let $\{\mathbf{x}(k)\}$ be generated by DEGAS. If T is pseudo-contractive with modulus $c\in(0,1)$ then*

$$\mathbb{E}[\|\mathbf{x}(k)-\mathbf{x}^\star\|^2]\leq\rho_{\mathcal{P}}^k\|\mathbf{x}(0)-\mathbf{x}^\star\|^2,\ \forall k\in\mathbb{N}_0,\tag{16}$$

where $\rho_{\mathcal{P}}=\alpha_{\mathcal{P}}\rho_a+(1-\alpha_{\mathcal{P}})\rho_c$ with

$$\alpha_{\mathcal{P}}=\frac{1}{1-\phi(\rho_a)/\phi(\rho_c)}\in(0,1).\tag{17}$$

In (17), $\phi(\rho_a)\geq 0$ and $\phi(\rho_c)<0$.

Proof. See Appendix C. □

The expectation in Theorem 2 is taken jointly over historical delays and block selection. A remarkable feature of Theorem 2 is that it allows to compute an explicit convergence rate bound for any given delay distribution $\mathcal{P}$. The convergence factor $\rho_{\mathcal{P}}$ is a convex combination of the corresponding quantities for the synchronous (centralized) and bounded-delay models, and the mixing parameter $\alpha_{\mathcal{P}}$ depends on the delay distribution $\mathcal{P}$. However, from (17), it is not straightforward to see how qualitative characteristics of the delay distribution (e.g., the mean or the variance) affect $\phi(\rho_c)$, $\phi(\rho_a)$, and $\alpha_{\mathcal{P}}$. As we will show next, such insight can be developed using the concept of stochastic dominance (Hadar & Russell, 1969).

Effect of delay under stochastic dominance: Suppose that $\mathcal{P}$ and $\mathcal{P}'$ are two probability distributions defined by (15).

Definition 3 (Stochastic dominance)**.** *We say that $\mathcal{P}$ first-order stochastically dominates $\mathcal{P}'$ ($\mathcal{P}\succeq_1\mathcal{P}'$) if*

$$\sum_{j=0}^{i}P_j\geq\sum_{j=0}^{i}P'_j,\quad\forall i\in\{0,\dots,\bar{\tau}\},$$

i.e., $\mathcal{P}$ always has a larger cumulative probability than $\mathcal{P}'$.

The stochastic dominance model compares the proportion of small delays in two delay distributions, which is different but has close connections to the mean-variance model:

Proposition 1. *Suppose that $\mathcal{P}\succeq_1\mathcal{P}'$. Then, the mean value of $\mathcal{P}$ is smaller than or equal to that of $\mathcal{P}'$ and if they share the same mean value, then the variance of $\mathcal{P}$ is smaller than or equal to that of $\mathcal{P}'$.*

Proof. These results are straightforward to derive from (Hadar & Russell, 1969). For sake of completeness, we provide an elementary proof in in Appendix D. □

The stochastic dominance concept allows us to prove the following attractive property of DEGAS:

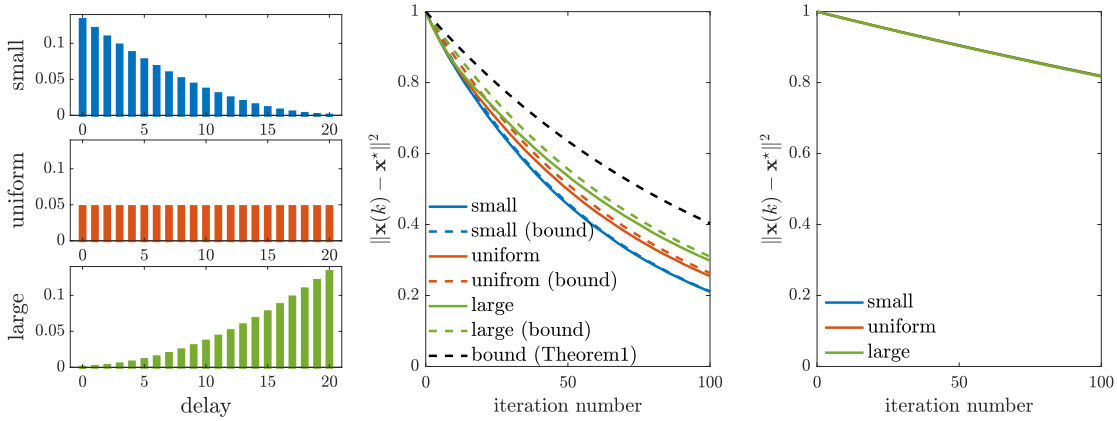
Lemma 3. *If $\mathcal{P}\succeq_1\mathcal{P}'$, then $\rho_{\mathcal{P}}\leq\rho_{\mathcal{P}'}$.*

Proof. See Appendix E. □

In particular, Lemma 3 demonstrates that an execution environment that delivers a larger portion of small delays will allow for a stronger convergence guarantee for DEGAS.

Demonstration with a simple operator: To demonstrate the self-adaptivity of DEGAS to actual delays, we consider the simple operator $\mathrm{T}(\mathbf{x})=0.8\mathbf{x}$ and three stochastic delay models. Note that when the delays are generated by stochastic models, the number of workers does not affect the update (6) or the convergence of DEGAS in terms of iteration index. We choose $m=20$, $\bar{\tau}=20$, and each block $x_i\in\mathbb{R}$. The stochastic models are: For any $i\in[0,\bar{\tau}]$, 1) **small**: $P_i=\frac{(\bar{\tau}+1-i)^2}{\sum_{j=1}^{\bar{\tau}+1}j^2}$; **uniform**: $P_i=\frac{1}{\bar{\tau}+1}$; **large**: $P_i=\frac{(i+1)^2}{\sum_{j=1}^{\bar{\tau}+1}j^2}$. If the generated value is larger than k, we set $\tau(k)=k$ to ensure $\tau(k)\in[0,k]$. We also run ARock with the same operator for comparison. For ARock (4), we fine tune $\gamma(k)$ in its theoretical range $(0,\frac{1}{2\bar{\tau}/\sqrt{m}+1})$ in (Peng et al., 2016) and draw blocks to update uniformly at random.

Figure 1. Adaptivity of DEGAS and ARock to real delays (small (bound), uniform (bound), and large (bound) are bounds in Theorem 2).



(a) delay distribution (b) DEGAS (c) ARock

We execute 2000 runs of both algorithms, each for 100 iterations, and plot the average result in Figure 1. We also plot the rate bounds in Theorems 1–2. We can see that DEGAS converges significantly faster when the proportion of small delays increases (large → uniform → small), demonstrating an excellent delay adaptivity of DEGAS. Moreover, the convergence bound in Theorem 2 is very tight for the simulated scenarios and accurately reflects the effect of delays on the DEGAS iterates. We also observe that Theorem 1, despite that it only uses the upper delay bound, is relatively tight. In contrast to DEGAS, no clear difference in the convergence speed of ARock under the three delay patterns can be observed due to the small step-size caused by the large $\bar{\tau}$. In addition, ARock is much slower than DEGAS for all three delay models. Note that the actual delays reported in (Mishchenko et al., 2022) have a much higher proportion of small delays than our **small** model, suggesting that the benefits of DEGAS could be even larger in practice.

2.2.2. Convergence on unbounded delay

We also study DEGAS under unbounded delays using the concept of total asynchrony (Bertsekas & Tsitsiklis, 1989).

Assumption 4 (Total asynchrony)**.** *The delay sequence* $\{\tau(k)\}$ *satisfies* $\lim_{k\to+\infty} k-\tau(k)=+\infty$.

Assumption 4 is very general and guarantees that old information must eventually be purged from the system.

Theorem 4. *Suppose that Assumptions 1,4 hold and* $\mathbf{x}^\star \in Fix\,\mathrm{T}$. *Let* $\{\mathbf{x}(k)\}$ *be generated by DEGAS. If* T *is averaged, then* $\lim_{k\to+\infty}\inf_{t\le k}\mathbb{E}\left[\|(\mathrm{Id}-\mathrm{T})(\mathbf{x}(t))\|\right]=0$.

Proof. See Appendix F. □

The expectation in Theorem 4 is taken over historical block selections. Under Assumption 4, it is impractical to derive explicit convergence rates due to the lack of bounds that limit how quickly the delays can grow large.

Below, we derive explicit convergence rates for DEGAS under the following delay model which satisfies Assumption 4 but adds a sublinearly or linearly growing delay bound.

Assumption 5 (Sublinear & linear delay growth bound)**.** *There exist* $\eta\in(0,1)$, $\beta\in(0,1]$, *and* $\gamma\ge 1$ *such that* $\tau(k)\le\eta k^\beta+\gamma\ \forall k\in\mathbb{N}_0$.

Theorem 5 (Sublinear convergence)**.** *Suppose that Assumption 5 holds. Let* $\{\mathbf{x}(k)\}$ *be generated by DEGAS. If* T *is pseudo-contractive with modulus* $c\in(0,1)$, *then*

$$\mathbb{E}[\|\mathbf{x}(k)-\mathbf{x}^\star\|^2]=\begin{cases}O\left(\left(1-\frac{1-c^2}{m}\right)^{k^{1-\beta}}\right), & \beta\in(0,1),\\ O(1/k), & \beta=1.\end{cases}$$

Proof. See Appendix G. □

Like Theorems 1, 4, the expectation in Theorem 5 is taken over historical block selections. Theorems 1 and 5 display how delays affect the order of convergence of DEGAS. Such a relationship is summarized in a more clear way in Table 1. In brief, delays that are allowed to grow large quicker lead to slower convergence, which coincides with intuition.

delay bound	constant	$O(k^\beta)$, $\beta\in(0,1)$	$O(k)$
rate	linear	$O(\rho_c^{k^{1-\beta}})$	$O(1/k)$

Table 1. Synchrony and convergence rate ($\rho_c=1-\frac{1-c^2}{m}$).

Wu et al. (2022b) derive similar results as Theorem 5 for the asynchronous BCD specialized from ARock and prove that they are optimal in terms of the convergence rate order. Hannah & Yin (2018) show that ARock converges under

certain stochastic and deterministic unbounded delay models. However, when considering deterministic delay models, neither of them guarantees the convergence of ARock under the more general total asynchrony assumption. Moreover, they both require carefully designed delay-dependent step-sizes to guarantee convergence, while DEGAS can converge with delay-free parameters.

3. Applications

By concretizing the operator T in DEGAS, we obtain novel and efficient asynchronous variants of BCD and ADMM.

3.1. Delay-agnostic asynchronous BCD

BCD (Richtárik & Takáč, 2014) solves the composite optimization problem

$$\underset{\mathbf{x}\in\mathbb{R}^d}{\text{minimize}}\ f(\mathbf{x})+\sum_{i=1}^m r_i(x_i), \tag{18}$$

where $f:\mathbb{R}^d\to\mathbb{R}$ is convex and L-smooth, $x_i\in\mathbb{R}^{d_i}$ is the ith block of $\mathbf{x}$, i.e., $\mathbf{x}=(x_1,\dots,x_m)$, and each $r_i:\mathbb{R}^{d_i}\to\mathbb{R}\cup\{+\infty\}$ is closed and convex. At each iteration k, BCD chooses one block $i\in[m]$ and updates

$$x_i(k+1)=\mathrm{prox}_{\gamma r_i}(x_i(k)-\gamma\nabla_i f(\mathbf{x}(k))), \tag{19}$$

while $x_j(k+1)=x_j(k)$ for all $j\neq i$. Here, $\gamma>0$ is a step-size and $\nabla_i f(\cdot)$ is the partial gradient of f with respect to x_i. This is equivalent to the coordinate update (2) with

$$\mathrm{T}_i(\mathbf{x})=\mathrm{prox}_{\gamma r_i}(x_i-\gamma\nabla_i f(\mathbf{x})),\ \forall i\in[m]. \tag{20}$$

We refer to DEGAS with T_i defined in (20) as delay-agnostic asynchronous BCD. Compared to the existing asynchronous BCD (Sun et al., 2017; Cheung et al., 2021) that is specialized from ARock, the delay-agnostic asynchronous BCD enjoys the same advantages as DEGAS over ARock, i.e., fast convergence under delay-free parameters.

If the optimal solution set of (18) is non-empty, so is $Fix\,\mathrm{T}$ where $\mathrm{T}=(\mathrm{T}_1,\dots,\mathrm{T}_m)$ with each T_i given by (20), and every $\mathbf{x}^\star\in Fix\,\mathrm{T}$ is an optimal solution of problem (18) (Bauschke et al., 2011). Under proper conditions, T is averaged and pseudo-contractive, which implies convergence of delay-agnostic asynchronous BCD by the results in § 2.2.

Lemma 6. *Suppose that $\gamma\in(0,2/L)$. The operator T defined in (20) is α-averaged with $\alpha=\frac{1}{\min(1,1/(L\gamma))+1/2}\in(0,1)$. If, in addition, f is μ-strongly convex for some $\mu\in(0,L]$, then T is pseudo-contractive with modulus $c=\sqrt{1-2\gamma\mu+\gamma^2\mu L}$.*

Proof. The claim follows by Theorem 25.8 in (Bauschke et al., 2011) and Proposition 5 in (Peng et al., 2016). □

Under the uniform random block selection rule and bounded delays, Sun et al. (2017); Cheung et al. (2021) establish convergence rates of the same order as Theorem 1 for the asynchronous BCD specialized from ARock. Neither of them requires Assumption 1, and Sun et al. (2017) also consider stochastic and deterministic unbounded delays and deterministic block selection rules. However, compared to our delay-agnostic asynchronous BCD, the asynchronous BCD (Sun et al., 2017; Cheung et al., 2021) inherits the disadvantages of ARock compared to DEGAS, i.e., delay-dependent step-sizes and the resulting slow convergence. Moreover, none of (Sun et al., 2017; Cheung et al., 2021) provide convergence results under Assumptions 4 or 5.

3.2. Delay-agnostic asynchronous ADMM

Consider the consensus optimization problem:

$$\underset{z\in\mathbb{R}^d}{\text{minimize}}\ \sum_{i=1}^m F_i(z), \tag{21}$$

where each F_i is convex and closed. Problem (21) includes many popular problems, including empirical risk minimization (Boyd et al., 2011). Letting $\mathbf{z}=(z_1,\dots,z_m)$, $F(\mathbf{z})=\sum_{i=1}^m F_i(z_i)$, and $\mathcal{C}=\{\mathbf{z}:z_1=\ldots=z_m\}$, problem (21) can be rewritten as

$$\underset{\mathbf{z}\in\mathbb{R}^{md}}{\text{minimize}}\ F(\mathbf{z})+\mathcal{I}_{\mathcal{C}}(\mathbf{z}), \tag{22}$$

where $\mathcal{I}_{\mathcal{C}}$ is the indicator function of $\mathcal{C}$. One popular way of solving (22) is to use the update (1) with T defined by the Douglas-Rachford splitting of $\partial\mathcal{I}_{\mathcal{C}}$ and ∂F, i.e.,

$$\mathrm{T}=\mathrm{Id}+\lambda(\mathrm{prox}_{\gamma F}\circ(2\,\mathrm{prox}_{\gamma\mathcal{I}_{\mathcal{C}}}-\mathrm{Id})-\mathrm{prox}_{\gamma\mathcal{I}_{\mathcal{C}}}), \tag{23}$$

where $\lambda\in(0,2)$. If the optimal solution set of (22) is non-empty, so is $Fix\,\mathrm{T}$ and, for any $\mathbf{x}^\star\in Fix\,\mathrm{T}$, $\mathrm{prox}_{\gamma\mathcal{I}_{\mathcal{C}}}(\mathbf{x}^\star)$ is an optimal solution of problem (22) (Bauschke et al., 2011). We refer to DEGAS with T in (23) as delay-agnostic asynchronous ADMM because its synchronous counterpart with $\lambda=1$ is equivalent to ADMM (see Appendix H).

The delay-agnostic asynchronous ADMM can be implemented as in Algorithm 1 where $\mathrm{T}_i(\mathbf{x}^w)$ is evaluated via

$$z_i=\frac{1}{m}\sum_{i=1}^m x_i^w, \tag{24}$$

$$\mathrm{T}_i(\mathbf{x}^w)=x_i^w+\lambda(\mathrm{prox}_{\gamma F_i}(2z_i-x_i^w)-z_i). \tag{25}$$

Below we show that T in (23) is an averaged operator under proper conditions. By the results in § 2.2, this implies that the delay-agnostic asynchronous ADMM converges under both bounded and unbounded delays.

Lemma 7. *The operator T in (23) is $\lambda/2$-averaged.*

Proof. See Appendix I. □

For some special examples of F_i's, e.g., each F_i is the indicator function of a subspace and $\lambda = 1$, the operator T in (23) becomes pseudo-contractive (Bauschke et al., 2014). In such cases, the delay-agnostic asynchronous ADMM can achieve linear convergence for bounded delays by Theorems 1–2, and sublinear convergence for unbounded delays by Theorem 5 in Appendix G. Similarly, delay-adaptivity is ensured by Theorem 2.

Remark 3. *A closely related asynchronous ADMM algorithm is developed in (Zhang & Kwok, 2014). It performs updates according to* (24)–(25) *with* $\lambda = 1$*, but assumes that the number of workers is identical to the number of blocks, and requires that each* z_i *is updated by worker* i*. To guarantee convergence, they assume that in every iteration all the workers have an equal probability to return their local variable to the master, which rarely holds in practice. Moreover, they only prove* $O(1/k)$ *convergence in terms of the running-average* $\frac{1}{k}\sum_{t=0}^{k-1}\mathbf{z}(t)$ *when the delays are bounded and have no convergence guarantee for the last iterate* $\mathbf{z}(k)$*. However, we prove convergence of the last iterate for bounded (Theorem 1) and unbounded delays (Theorem 4). Such rates can be improved as discussed below Lemma 7 when the operator is pseudo-contractive.*

3.2.1. Extension to a more general problem

We extend the delay-agnostic asynchronous ADMM to solve

$$\underset{\mathbf{z}\in\mathcal{Z}}{\text{minimize}}\ f(\mathbf{z}) + r(\mathbf{z}), \tag{26}$$

where $\mathbf{z} = (z_1, \ldots, z_m)$ with each $z_i \in \mathbb{R}^{d_i}$, $\mathcal{Z} \subseteq \mathbb{R}^{\sum_{i=1}^m d_i}$ is a convex and closed set endowed with a simple projection operator, f is convex and L-smooth, and $r(\mathbf{z}) = \sum_{i=1}^m r_i(z_i)$ with each r_i being a convex, closed, but possibly non-smooth function. Some examples of $\mathcal{Z}$ include: i) When $\mathcal{Z} = \mathbb{R}^{\sum_{i=1}^m d_i}$, (26) reduces to problem (18); ii) When $\mathcal{Z} = \{\mathbf{z} : z_1 = z_2 = \ldots = z_m\}$, the problem (26) becomes consensus optimization, slightly more general than (22) since the objective function is allowed to have a non-separable smooth component; iii) When $\mathcal{Z} = \{\mathbf{z} : A\mathbf{z} \le b\}$ or $\mathcal{Z} = \{\mathbf{z} : A\mathbf{z} = b\}$ for a matrix A and a vector b, it becomes a resource allocation problem (Lin et al., 2015).

To exploit the composite structure of the objective function, we replace $\text{prox}_{\gamma F}$ in (23) by

$$\mathrm{T}' = \theta\,\text{prox}_{\gamma r}\circ(\mathrm{Id} - \gamma\nabla f) + (1-\theta)\,\mathrm{Id}, \tag{27}$$

where $\gamma \in (0, 2/L)$ and $\theta = \frac{\min(1, 1/(L\gamma))+1/2}{2}$. We average the proximal gradient operator with Id to make T' a $1/2$-averaged operator, which will be used in our convergence analysis (see Lemma 8 below). We also replace $\mathcal{C}$ in (23) by $\mathcal{Z}$. Then, the new operator takes the form

$$\mathrm{T} = \mathrm{Id} + \lambda(\mathrm{T}'\circ(2\,\text{prox}_{\gamma\mathcal{I}_\mathcal{Z}} - \mathrm{Id}) - \text{prox}_{\gamma\mathcal{I}_\mathcal{Z}}) \tag{28}$$

for $\lambda \in (0, 2)$, which can be simplified to

$$\begin{aligned}\mathrm{T} = &\,\text{prox}_{\gamma r}\circ(\mathrm{Id} - \gamma\nabla f)\circ(2\,\text{prox}_{\gamma\mathcal{I}_\mathcal{Z}} - \mathrm{Id})\\ &+ \frac{2}{3}(\mathrm{Id} - \text{prox}_{\gamma\mathcal{I}_\mathcal{Z}})\end{aligned} \tag{29}$$

when $\gamma = 1/L$ and $\lambda = 4/3$. If the optimal solution set of (26) is non-empty, so is $Fix\,\mathrm{T}$ with T in (28) and for any $\mathbf{x}^\star \in Fix\,\mathrm{T}$, $\text{prox}_{\gamma\mathcal{I}_\mathcal{Z}}(\mathbf{x}^\star)$ is an optimal solution of problem (26) (Bauschke et al., 2011).

We refer to DEGAS with T in (28) as extended delay-agnostic asynchronous ADMM, whose asynchronous implementation follows Algorithm 1 with $\mathrm{T}_i(\mathbf{x}^w)$ in Step 6 being evaluated via

$$z_i = [\text{prox}_{\gamma\mathcal{I}_\mathcal{Z}}(\mathbf{x}^w)]_i, \tag{30}$$

$$y_i = 2z_i - x_i^w, \tag{31}$$

$$\begin{aligned}\mathrm{T}_i(\mathbf{x}^w) = &\,\lambda\theta\,\text{prox}_{\gamma r_i}(y_i - \gamma\nabla_i f(y_i))\\ &+ (1-(1-\theta)\lambda)x_i^w - \lambda(1-2(1-\theta))z_i.\end{aligned} \tag{32}$$

Lemma 8. *The operator* T' *in* (27) *is* $1/2$*-averaged and* T *in* (28) *is* $\lambda/2$*-averaged.*

Proof. See Appendix J. □

With Lemma 8, convergence of the extended asynchronous ADMM for bounded and unbounded delays can be recovered from Theorems 1 and 4 in a straightforward manner.

4. Experiments

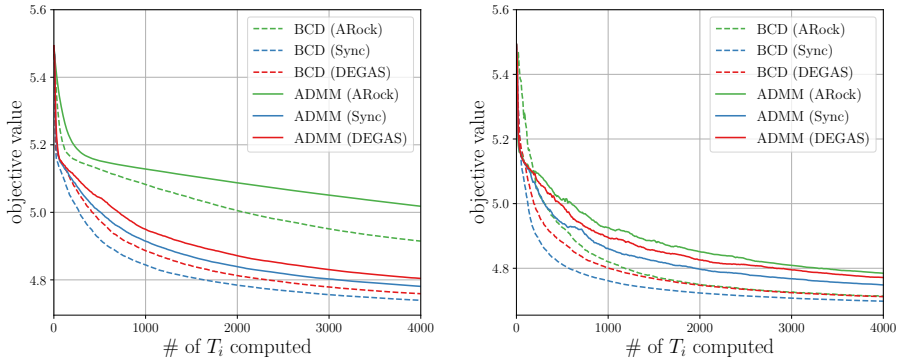
We evaluate the practical performance of DEGAS on Lasso and regularized logistic regression problems on the CIFAR100 dataset (Krizhevsky et al., 2009).

Let a_i be the feature of the ith sample, b_i be the corresponding label, and N be the number of samples. Then our test problems are on the form

$$\underset{x\in\mathbb{R}^d}{\text{minimize}}\ \frac{1}{N}\sum_{i=1}^N \ell_i(x) + \lambda_1\|x\|_1, \tag{33}$$

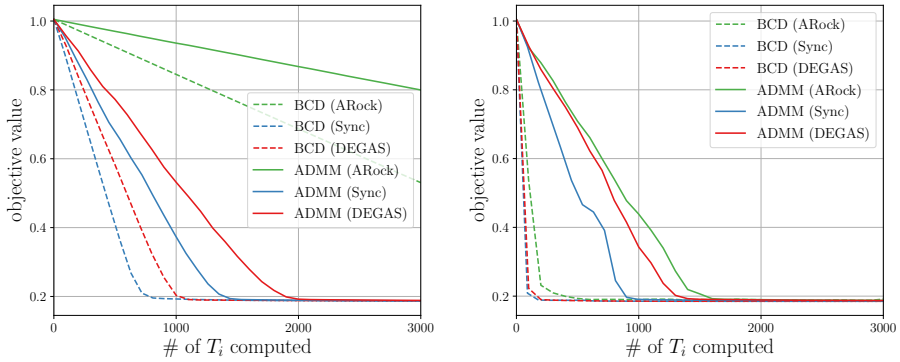
where $\ell_i(x) = \frac{1}{2}\|a_i x - b_i\|^2$ in Lasso and $\ell_i(x) = \log(1 + e^{-b_i(a_i^T x)}) + \frac{\lambda_2}{2}\|x\|^2$ in regularized logistic regression. We use $\lambda_1 = 10^{-3}$ and $\lambda_2 = 10^{-4}$. The proposed DEGAS algorithm is compared with ARock and its synchronous counterpart. In these methods, we choose the operator T as (20) with $\gamma = 1/L$ in BCD and (29) in the extended ADMM. We set $m = 20$ and implement all the methods on a 10-core machine (1 master and 9 workers) using the

Figure 2. Convergence for Lasso



(a) theoretical parameters (b) hand-tuned parameters

Figure 3. Convergence for Logistic regression



(a) theoretical parameters (b) hand-tuned parameters

message-passing framework MPI4py (Dalcín et al., 2008). Note that we do not assume any delay model and that all delays are generated by real interactions between the master and workers. We consider both theoretical and hand-tuned parameters. In the former setting, we fine-tune the step-size of ARock within its theoretical range in (Feyzmahdavian & Johansson, 2021) which is broader than that in (Peng et al., 2016) in the settings of this experiment, while the other two methods have no parameters to tune. In the hand-tune step-size setting, we run all the methods for finding the fixed point of $\mathrm{Id}+\lambda(\mathrm{T}-\mathrm{Id})$, $\lambda>0$ and tune λ.

We plot the convergence in terms of the number of computed T_i in Figures 2–3, from which we make the following observations: 1) For both theoretical and hand-tuned step-sizes, DEGAS is much faster than ARock in all tested scenarios. Hence, not only is it easier to tune, but it also converges faster; 2) the synchronous method outperforms DEGAS in terms of the number of T_i's computed. However, as asynchronous methods can complete more computations within a fixed time interval compared to synchronous methods, DEGAS may converge faster in terms of wall-clock time, cf. the discussion in Appendix L. We also observe that DEGAS and the synchronous methods can converge with much larger hand-tuned step-sizes than ARock. We plot the delay distribution generated by the experiments in Appendix K.

5. Conclusion

We have proposed a delay-agnostic asynchronous coordinate update method (DEGAS) to find fixed-points of operators, a problem with broad applications in algebra, optimization, and game theory. Compared to ARock, which only converges under a delay-dependent parameter condition, the parameters of DEGAS can be determined without any knowledge of the delays. Moreover, DEGAS adapts well to the actual delays in the system and converges significantly faster than ARock using both theoretically justified and hand-tuned step-sizes in our numerical experiments.

Acknowledgements

This work was supported by WASP and the Swedish Research Council (Vetenskapsrådet) under grants 2019-05319 and 2020-03607. We thank the anonymous reviewers for their detailed and valuable feedback.

References

Agarwal, A. and Duchi, J. C. Distributed delayed stochastic optimization. *Advances in neural information processing systems*, 24, 2011.

Aviv, R. Z., Hakimi, I., Schuster, A., and Levy, K. Y. Asynchronous distributed learning : Adapting to gradient delays without prior knowledge. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139, pp. 436–445, 2021.

Aybat, N., Wang, Z., and Iyengar, G. An asynchronous distributed proximal gradient method for composite convex optimization. In *International Conference on Machine Learning*, pp. 2454–2462. PMLR, 2015.

Aytekin, A., Feyzmahdavian, H. R., and Johansson, M. Analysis and implementation of an asynchronous optimization algorithm for the parameter server. *arXiv preprint arXiv:1610.05507*, 2016.

Bauschke, H. H., Combettes, P. L., et al. *Convex analysis and monotone operator theory in Hilbert spaces*, volume 408. Springer, 2011.

Bauschke, H. H., Bello Cruz, J., Nghia, T. T., Phan, H. M., and Wang, X. The rate of linear convergence of the douglas–rachford algorithm for subspaces is the cosine of the friedrichs angle. *Journal of Approximation Theory*, 185:63–79, 2014.

Bertsekas, D. P. Distributed asynchronous computation of fixed points. *Mathematical Programming*, 27(1):107–120, 1983.

Bertsekas, D. P. and Tsitsiklis, J. N. Parallel and distributed computation: numerical methods. Prentice-Hall, Inc., 1989.

Boyd, S., Parikh, N., Chu, E., Peleato, B., Eckstein, J., et al. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends® in Machine learning*, 3(1):1–122, 2011.

Cheung, Y. K., Cole, R., and Tao, Y. Fully asynchronous stochastic coordinate descent: a tight lower bound on the parallelism achieving linear speedup. *Mathematical Programming*, 190:615–677, 2021.

Cohen, A., Daniely, A., Drori, Y., Koren, T., and Schain, M. Asynchronous stochastic optimization robust to arbitrary delays. *Advances in Neural Information Processing Systems*, 34:9024–9035, 2021.

Dalcín, L., Paz, R., Storti, M., and D'Elía, J. Mpi for python: Performance improvements and mpi-2 extensions. *Journal of Parallel and Distributed Computing*, 68 (5):655–662, 2008.

Feyzmahdavian, H. R. and Johansson, M. Asynchronous iterations in optimization: New sequence results and sharper algorithmic guarantees. *arXiv preprint arXiv:2109.04522*, 2021.

Feyzmahdavian, H. R., Aytekin, A., and Johansson, M. A delayed proximal gradient method with linear convergence rate. In *2014 IEEE International Workshop on Machine Learning for Signal Processing (MLSP)*, pp. 1–6. IEEE, 2014.

Frommer, A. Generalized nonlinear diagonal dominance and applications to asynchronous iterative methods. *Journal of Computational and Applied Mathematics*, 38(1): 105–124, 1991.

Glasgow, M. R. and Wootters, M. Asynchronous distributed optimization with stochastic delays. In *International Conference on Artificial Intelligence and Statistics*, pp. 9247–9279, 2022.

Hadar, J. and Russell, W. R. Rules for ordering uncertain prospects. *The American economic review*, 59(1):25–34, 1969.

Hale, M. T., Nedić, A., and Egerstedt, M. Asynchronous multiagent primal-dual optimization. *IEEE Transactions on Automatic Control*, 62(9):4421–4435, 2017.

Hannah, R. and Yin, W. On unbounded delays in asynchronous parallel fixed-point algorithms. *Journal of Scientific Computing*, 76(1):299–326, 2018.

Iutzeler, F., Malick, J., and de Oliveira, W. Asynchronous level bundle methods. *Mathematical Programming*, 184 (1):319–348, 2020.

Koloskova, A., Stich, S. U., and Jaggi, M. Sharper convergence guarantees for asynchronous sgd for distributed and federated learning. In *Advances in Neural Information Processing Systems*, 2022.

Krizhevsky, A., Hinton, G., et al. Learning multiple layers of features from tiny images. 2009.

Leblond, R., Pedregosa, F., and Lacoste-Julien, S. Improved asynchronous parallel optimization analysis for stochastic incremental methods. *Journal of Machine Learning Research*, 2018.

Li, M., Zhou, L., Yang, Z., Li, A., Xia, F., Andersen, D. G., and Smola, A. Parameter server for distributed machine learning. In *Big Learning NIPS Workshop*, volume 6, pp. 2, 2013.

Lian, X., Zhang, W., Zhang, C., and Liu, J. Asynchronous decentralized parallel stochastic gradient descent. In *International Conference on Machine Learning*, pp. 3043–3052. PMLR, 2018.

Lin, T., Ma, S., and Zhang, S. On the global linear convergence of the ADMM with multiblock variables. *SIAM Journal on Optimization*, 25(3):1478–1497, 2015.

Liu, J. and Wright, S. J. Asynchronous stochastic coordinate descent: Parallelism and convergence properties. *SIAM Journal on Optimization*, 25(1):351–376, 2015.

Liu, J., Wright, S., Ré, C., Bittorf, V., and Sridhar, S. An asynchronous parallel stochastic coordinate descent algorithm. In *International Conference on Machine Learning*, pp. 469–477. PMLR, 2014.

Luo, Q., He, J., Zhuo, Y., and Qian, X. Prague: High-performance heterogeneity-aware asynchronous decentralized training. In *Proceedings of the Twenty-Fifth International Conference on Architectural Support for Programming Languages and Operating Systems*, pp. 401–416, 2020.

Mania, H., Pan, X., Papailiopoulos, D., Recht, B., Ramchandran, K., and Jordan, M. I. Perturbed iterate analysis for asynchronous stochastic optimization. *SIAM Journal on Optimization*, 27(4):2202–2229, 2017.

Mehyar, M., Spanos, D., Pongsajapan, J., Low, S. H., and Murray, R. M. Asynchronous distributed averaging on communication networks. *IEEE/ACM Transactions On Networking*, 15(3):512–520, 2007.

Mishchenko, K., Iutzeler, F., Malick, J., and Amini, M.-R. A delay-tolerant proximal-gradient algorithm for distributed learning. In *International Conference on Machine Learning*, pp. 3587–3595. PMLR, 2018.

Mishchenko, K., Bach, F., Even, M., and Woodworth, B. Asynchronous SGD beats minibatch SGD under arbitrary delays. In *Advances in Neural Information Processing Systems*, 2022.

Moallemi, C. C. and Van Roy, B. Convergence of min-sum message-passing for convex optimization. *IEEE Transactions on Information Theory*, 56(4):2041–2050, 2010.

Nesterov, Y. Efficiency of coordinate descent methods on huge-scale optimization problems. *SIAM Journal on Optimization*, 22(2):341–362, 2012.

Peng, Z., Xu, Y., Yan, M., and Yin, W. ARock: an algorithmic framework for asynchronous parallel coordinate updates. *SIAM Journal on Scientific Computing*, 38(5): A2851–A2879, 2016.

Recht, B., Re, C., Wright, S., and Niu, F. Hogwild!: A lock-free approach to parallelizing stochastic gradient descent. *Advances in Neural Information Processing Systems*, 24: 693–701, 2011.

Richtárik, P. and Takáč, M. Iteration complexity of randomized block-coordinate descent methods for minimizing a composite function. *Mathematical Programming*, 144(1): 1–38, 2014.

Richtárik, P. and Takáč, M. Distributed coordinate descent method for learning with big data. *The Journal of Machine Learning Research*, 17(1):2657–2681, 2016.

Soori, S., Mishchenko, K., Mokhtari, A., Dehnavi, M. M., and Gurbuzbalaban, M. DAve-QN: A distributed averaged quasi-newton method with local superlinear convergence rate. In *International Conference on Artificial Intelligence and Statistics*, pp. 1965–1976. PMLR, 2020.

Sra, S., Yu, A. W., Li, M., and Smola, A. Adadelay: Delay adaptive distributed stochastic optimization. In *Artificial Intelligence and Statistics*, pp. 957–965. PMLR, 2016.

Sun, T., Hannah, R., and Yin, W. Asynchronous coordinate descent under more realistic assumption. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pp. 6183–6191, 2017.

Sun, T., Sun, Y., Li, D., and Liao, Q. General proximal incremental aggregated gradient algorithms: Better and novel results under general scheme. *Advances in Neural Information Processing Systems*, 32:996–1006, 2019.

Wright, S. J. Coordinate descent algorithms. *Mathematical Programming*, 151(1):3–34, 2015.

Wu, T., Yuan, K., Ling, Q., Yin, W., and Sayed, A. H. Decentralized consensus optimization with asynchrony and delays. *IEEE Transactions on Signal and Information Processing over Networks*, 4(2):293–307, 2017.

Wu, X., Magnússon, S., Feyzmahdavian, H. R., and Johansson, M. Delay-adaptive step-sizes for asynchronous learning. In *International Conference on Machine Learning*, pp. 24093–24113. PMLR, 2022a.

Wu, X., Magnússon, S., Feyzmahdavian, H. R., and Johansson, M. Optimal convergence rates of totally asynchronous optimization. In *2022 IEEE 61st Conference on Decision and Control (CDC)*, pp. 6484–6490. IEEE, 2022b.

Zhang, J. and You, K. Asyspa: An exact asynchronous algorithm for convex optimization over digraphs. *IEEE Transactions on Automatic Control*, 65(6):2494–2509, 2019.

Zhang, R. and Kwok, J. Asynchronous distributed ADMM for consensus optimization. In *International Conference on Machine Learning*, pp. 1701–1709. PMLR, 2014.

Zhou, Z., Mertikopoulos, P., Bambos, N., Glynn, P., Ye, Y., Li, L.-J., and Fei-Fei, L. Distributed asynchronous optimization with unbounded delays: How slow can you go? In *International Conference on Machine Learning*, pp. 5970–5979. PMLR, 2018.

A. Proof of Theorem 1

At each iteration $k \in \mathbb{N}_0$, each $i \in [m]$ has equal probability to be selected as $i(k)$. Then by (6),

$$\begin{aligned}
&\mathbb{E}[\|\mathbf{x}(k+1) - \mathbf{x}^\star\|^2] \\
=&\sum_{i=1}^{m} \frac{1}{m} \left(\| \mathrm{T}_i(\mathbf{x}(k-\tau(k))) - x_i^\star\|^2 + \sum_{j\neq i} \|x_j(k) - x_j^\star\|^2 \right) \\
=&\frac{1}{m}\| \mathrm{T}(\mathbf{x}(k-\tau(k))) - \mathbf{x}^\star\|^2 + \left(1 - \frac{1}{m}\right) \|\mathbf{x}(k) - \mathbf{x}^\star\|^2,
\end{aligned} \tag{34}$$

where $x_i^\star \in \mathbb{R}^{d_i}$ is the ith block of $\mathbf{x}^\star$ and the expectation is taken over the block $i(k)$. Taking (34) at hand, we are ready to prove the results for both averaged and pseudo-contractive operator T.

A.1. Proof for averaged T

For all $k \in \mathbb{N}_0$, let

$$\begin{aligned}
V(k) &= \mathbb{E}[\|\mathbf{x}(k) - \mathbf{x}^\star\|^2], \\
W(k) &= \frac{1-\alpha}{m\alpha}\mathbb{E}[\|(\mathrm{Id} - \mathrm{T})(\mathbf{x}(k))\|^2],
\end{aligned}$$

where the expectations are taken over the historical updated blocks $\{i(t)\}_{t\leq k}$. For all $t \in \mathbb{N}_0$, define

$$\begin{aligned}
\mathcal{A}(t) &= [t(\bar{\tau}+1), (t+1)(\bar{\tau}+1)), \\
a(t) &= \arg\ \max\nolimits_{k\in\mathcal{A}(t)} V(k).
\end{aligned}$$

The proof includes three steps. Step 1 establishes the relationship between $\{V(k)\}$ and $\{W(k)\}$:

$$V(k+1) \leq \max_{\max(0,k-\bar{\tau})\leq \ell \leq k} V(\ell) - W(k-\tau(k)), \quad \forall k \in \mathbb{N}_0. \tag{35}$$

Based on (35), step 2 proves

$$V(a(t)) \leq V(a(t-1)) - W(a(t) - 1 - \tau(a(t)-1)), \quad \forall t \in \mathbb{N}, \tag{36}$$

which is then used to derive (9) in step 3.

Step 1: The proof uses Proposition 4.25 in (Bauschke et al., 2011): Denote the average parameter of T as $\alpha \in (0,1)$. Then, for any $\mathbf{x}, \mathbf{y} \in \mathbb{R}^d$,

$$\| \mathrm{T}(\mathbf{x}) - \mathrm{T}(\mathbf{y})\|^2 \leq \|\mathbf{x} - \mathbf{y}\|^2 - \frac{1-\alpha}{\alpha}\|(\mathrm{Id} - \mathrm{T})(\mathbf{x}) - (\mathrm{Id} - \mathrm{T})(\mathbf{y})\|^2. \tag{37}$$

Substituting $\mathbf{x} = \mathbf{x}(k-\tau(k))$ and $\mathbf{y} = \mathbf{x}^\star$ into (37) and using $\mathrm{T}(\mathbf{x}^\star) = \mathbf{x}^\star$, we have

$$\begin{aligned}
&\| \mathrm{T}(\mathbf{x}(k-\tau(k))) - \mathbf{x}^\star\|^2 \\
\leq&\|\mathbf{x}(k-\tau(k)) - \mathbf{x}^\star\|^2 - \frac{1-\alpha}{\alpha}\|(\mathrm{Id} - \mathrm{T})(\mathbf{x}(k-\tau(k)))\|^2.
\end{aligned} \tag{38}$$

Substituting (38) into (34) and taking expectation on both sides of the resulting equation yields

$$\begin{aligned}
V(k+1) \leq&\frac{V(k-\tau(k))}{m} + \left(1 - \frac{1}{m}\right) V(k) - W(k-\tau(k)) \\
\leq& \max_{\max(0,k-\tau(k))\leq \ell \leq k} V(\ell) - W(k-\tau(k)).
\end{aligned} \tag{39}$$

By (39) and $\tau(k) \leq \bar{\tau}$ assumed in Assumption 2, we have (35).

Step 2: We first show by induction that for any $k \in \mathbb{N}_0$ satisfying $k+1 \in \mathcal{A}(t)$ or equivalently, $k \in [t(\bar{\tau}+1)-1, (t+1)(\bar{\tau}+1)-1)$,

$$\max_{\max(0,k-\bar{\tau})\le \ell \le k} V(\ell) \le \max_{\ell \in \mathcal{A}(t-1)} V(\ell). \tag{40}$$

When $k = t(\bar{\tau}+1)-1$, since $[\max(0, k-\bar{\tau}), k] = \mathcal{A}(t-1)$, the equation (40) holds. Suppose that (40) holds at $k = k'-1$ for some k' satisfying $k'+1 \in \mathcal{A}(t)$. Then, by letting $k = k'-1$ in both (35) and (40), we have

$$\begin{aligned} V(k') &\overset{(35)}{\le} \max_{\max(0,k'-1-\bar{\tau})\le \ell \le k'-1} V(\ell) - W(k'-1-\tau(k'-1)) \\ &\le \max_{\max(0,k'-1-\bar{\tau})\le \ell \le k'-1} V(\ell) \\ &\overset{(40)}{\le} \max_{\ell \in \mathcal{A}(t-1)} V(\ell). \end{aligned}$$

This, together with (40) at $k = k'-1$, yields (40) at $k = k'$. Following this induction procedure we obtain (40) for all k satisfying $k+1 \in \mathcal{A}(t)$. Then, by letting $k = a(t)-1$ in both (35) and (40), we have

$$\begin{aligned} V(a(t)) &\overset{(35)}{\le} \max_{\max(0,a(t)-1-\bar{\tau})\le \ell \le a(t)-1} V(\ell) - W(a(t)-1-\tau(a(t)-1)) \\ &\overset{(40)}{\le} \max_{\ell \in \mathcal{A}(t-1)} V(\ell) - W(a(t)-1-\tau(a(t)-1)) \\ &= V(a(t-1)) - W(a(t)-1-\tau(a(t)-1)), \end{aligned}$$

i.e., (36) holds.

Step 3: For any $t' \in \mathbb{N}$, by adding (36) from $t=1$ to $t=t'$, we have

$$\begin{aligned} \sum_{t=1}^{t'} W(a(t)-1-\tau(a(t)-1)) &\le V(a(0)) - V(a(t')) \\ &\le V(a(0)) \\ &= V(0), \end{aligned} \tag{41}$$

where $V(a(0)) = V(0)$ can be easily derived from (35). For any $k \in \mathbb{N}_0$, let $t'(k) = \lfloor k/(\bar{\tau}+1) \rfloor - 1$. Then, for all $t \in [t'(k)]$,

$$\begin{aligned} a(t)-1-\tau(a(t)-1) &\le a(t) \\ &\le a(t'(k)) \\ &\le (t'(k)+1)(\bar{\tau}+1) \\ &\le k, \end{aligned} \tag{42}$$

which, together with (41), yields

$$\begin{aligned} \min_{1\le \ell \le k} W(\ell) &\overset{(42)}{\le} \min_{1\le t \le t'(k)} W(a(t)-1-\tau(a(t)-1)) \\ &\le \frac{1}{t'(k)} \sum_{t=1}^{t'(k)} W(a(t)-1-\tau(a(t)-1)) \\ &\overset{(41)}{\le} \frac{V(0)}{t'(k)}. \end{aligned}$$

Moreover, $t'(k) \ge k/(\bar{\tau}+1) - 2$. Then, we have

$$\begin{aligned} \min_{1\le \ell \le k} W(\ell) &\le \frac{V(0)}{k/(\bar{\tau}+1)-2} \\ &= O(1/k), \end{aligned}$$

i.e., (9) holds.

A.2. Proof for pseudo-contractive T

By the pseudo-contractivity of T,

$$\|\,\mathrm{T}(\mathbf{x}(k-\tau(k)))-\mathbf{x}^\star\|^2 \le c^2\|\mathbf{x}(k-\tau(k))-\mathbf{x}^\star\|^2,$$

substituting which into (34) and taking expectation on both sides of the resulting equation ensures

$$V(k+1) \le \frac{c^2}{m}V(k-\tau(k)) + \left(1-\frac{1}{m}\right)V(k). \tag{43}$$

To proceed, we establish a sequence result in the following lemma.

Lemma 9. *Suppose that the following holds for a non-negative sequence $\{V(k)\}$ and two positive constants $p,q\in(0,1)$ satisfying $p+q<1$:*

$$V(k+1) \le pV(k) + q\max_{\max(0,k-\bar\tau)\le\ell\le k} V(\ell). \tag{44}$$

Then,

$$V(k) \le \rho^k V(0),\ \forall k\in\mathbb{N}_0, \tag{45}$$

where

$$\rho = (p+q)^{\frac{1}{1+(1-p)\bar\tau}}. \tag{46}$$

Proof. We prove (45) by induction. Clearly, (45) holds for $k=0$. Suppose that (45) holds for all $k\in[0,k']$ for some $k'\in\mathbb{N}_0$. Then, by (44),

$$V(k'+1) \le (p\rho^{k'} + q\rho^{k'-\bar\tau})V(0).$$

Hence, to show (45) at $k=k'+1$, it suffices to prove $p\rho^{k'} + q\rho^{k'-\bar\tau} \le \rho^{k'+1}$, which is equivalent to

$$p + q\rho^{-\bar\tau} \le \rho. \tag{47}$$

Therefore, if (47) holds, so does (45) at $k=k'+1$. Following this induction procedure, we will have that (45) holds for all $k\in\mathbb{N}_0$.

Next, we prove (47), which includes three steps. Step 1 shows the equivalence between (47) and

$$\left(\frac{p+q}{\rho}\right)^{\frac{1}{1-p}} \ge \frac{q}{\rho-p}. \tag{48}$$

Step 2 proves the following inequality: For any $\alpha\in[p+q,1]$,

$$\left(\frac{p+q}{\alpha}\right)^{\frac{1}{1-p}} \ge \frac{q}{\alpha-p}. \tag{49}$$

Step 3 combines the first two steps and derives (47).

Step 1: The equation (47) is equivalent to

$$\rho^{\bar\tau} \ge \frac{q}{\rho-p}. \tag{50}$$

By (46),

$$\begin{aligned}
\rho^{\bar\tau} &= (p+q)^{\frac{\bar\tau}{1+(1-p)\bar\tau}} \\
&= (p+q)^{\frac{(1-p)\bar\tau}{1+(1-p)\bar\tau}\cdot\frac{1}{1-p}} \\
&= (p+q)^{\left(1-\frac{1}{1+(1-p)\bar\tau}\right)\cdot\frac{1}{1-p}} \\
&= \left((p+q)^{\left(1-\frac{1}{1+(1-p)\bar\tau}\right)}\right)^{\frac{1}{1-p}} \\
&= \left(\frac{p+q}{(p+q)^{\frac{1}{1+(1-p)\bar\tau}}}\right)^{\frac{1}{1-p}} \\
&\overset{(46)}{=} \left(\frac{p+q}{\rho}\right)^{\frac{1}{1-p}}.
\end{aligned}$$

Therefore, (50) is equivalent to (48), which, together with the equivalence between (47) and (50), yields the equivalence between (47) and (48).

Step 2: By Bernoulli's inequality, for any $a \in (0,1],\ b \geq 1$,

$$\begin{aligned} a^b &= (1-(1-a))^b \\ &\geq 1-(1-a)b. \end{aligned} \tag{51}$$

Letting $a = \frac{p+q}{\alpha}$ and $b = \frac{1}{1-p}$, we have

$$\left(\frac{p+q}{\alpha}\right)^{\frac{1}{1-p}} \geq 1 - \frac{1-(p+q)/\alpha}{1-p}.$$

Therefore, (49) holds if

$$1 - \frac{1-(p+q)/\alpha}{1-p} \geq \frac{q}{\alpha - p}, \tag{52}$$

which is equivalent to $h(\alpha) := p\alpha^2 - (1+p+q)p\alpha + p(p+q) \leq 0$. Note that

$$h(p+q) = 0, \quad h(1) = 0.$$

Let $\gamma = \frac{1-\alpha}{1-p-q}$ which satisfies $\gamma \in [0,1]$ due to $\alpha \in [p+q, 1]$. By the convexity of $h(\alpha)$, we have

$$\begin{aligned} h(\alpha) &= h(\gamma(p+q) + (1-\gamma)) \\ &\leq \gamma h(p+q) + (1-\gamma)h(1) \\ &= 0, \end{aligned}$$

which further indicates (52) and (49).

Step 3: By (49) with $\alpha = \rho$, we obtain (48). Then, by the equivalence between (47) and (49), the equation (47) holds, which concludes the proof.

□

Since $\tau(k) \leq \bar{\tau}$, (43) yields (44) with $p = 1 - \frac{1}{m}$ and $q = \frac{c^2}{m}$. Then by Lemma 9, we obtain (10). This completes the proof.

Remark on Lemma 9: The linear rate (10) in Lemma 9 significantly improves the existing rate in (Feyzmahdavian & Johansson, 2021) in the sense of tightness. Specifically, (Feyzmahdavian & Johansson, 2021) proves that for any non-negative sequence $\{V(k)\}$ satisfying (44), (45) holds with

$$\rho = (p+q)^{\frac{1}{\bar{\tau}+1}}. \tag{53}$$

To distinguish ρ in (46) and (53), we denote the former as ρ_1 and the latter as ρ_2. Since $p+q \in (0,1)$ and $\frac{1}{1+(1-p)\bar{\tau}} \geq \frac{1}{1+\bar{\tau}}$, it always holds that $\rho_1 \leq \rho_2$. The difference between ρ_1 and ρ_2 becomes clear when we look at their resulting iteration complexities, where a smaller iteration complexity indicates a tighter convergence rate bound. To guarantee $V(k) \leq \epsilon$ for some $\epsilon > 0$, (45) with $\rho = \rho_1$ and $\rho = \rho_2$ requires

$$\begin{aligned} K_1(\epsilon) &= \frac{(1+(1-p)\bar{\tau})\ln\frac{V(0)}{\epsilon}}{\ln\frac{1}{p+q}}, \\ K_2(\epsilon) &= \frac{(1+\bar{\tau})\ln\frac{V(0)}{\epsilon}}{\ln\frac{1}{p+q}}, \end{aligned}$$

respectively. Here, we can see that

$$\frac{K_1(\epsilon)}{K_2(\epsilon)} = \frac{1+(1-p)\bar{\tau}}{1+\bar{\tau}},$$

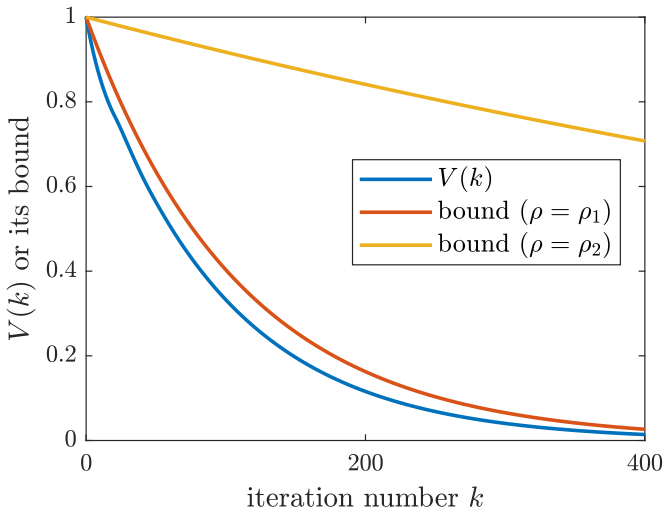


Figure 4. Tightness of rate bounds in Lemma 9 ($\rho = \rho_1$) and (Feyzmahdavian & Johansson, 2021) ($\rho = \rho_2$).

which can be very small when p is close to 1 and $\bar{\tau}$ is large. For example, for equation (43) which yields (44) with $p = 1 - \frac{1}{m}$, if we let $\bar{\tau} = m = 20$, then

$$\frac{K_1(\epsilon)}{K_2(\epsilon)} = \frac{1 + \bar{\tau}/m}{1 + \bar{\tau}} = \frac{2}{21},$$

indicating that the rate (45) yield by $\rho = \rho_1$ is much tighter than that yield by $\rho = \rho_2$.

We also visualize the tightness of the two bounds by considering the convergence of a concrete sequence $\{V(k)\}$: $V(0) = 1$. At each $k \in \mathbb{N}_0$, (43) holds with equality, where $m = \bar{\tau} = 20$, $c = 0.8$, and $\tau(k) = \min(k, \bar{\tau})$. For this example, (44) holds with $p = 1 - \frac{1}{m} = 0.95$ and $q = c^2/m = 0.032$.

The convergence of $\{V(k)\}$ and its theoretical two bounds are displayed in Figure A.2. By A.2 we can see that the rate bound in Lemma 9 is close to the practical convergence of $\{V(k)\}$ and is much tighter than the rate bound in (Feyzmahdavian & Johansson, 2021).

B. Proof of (14) in Remark 2

The proof mainly uses Bernoulli's inequality: For any $a \in (0, 1)$ and $b \in (0, 1]$,

$$a^b \leq 1 - b(1 - a). \tag{54}$$

Letting $a = 1 - \frac{1-c^2}{m} = \rho_a^{1+\bar{\tau}/m}$ and $b = \frac{1}{1+6\left(\frac{\bar{\tau}}{m} + \sqrt{\frac{\bar{\tau}}{m}}\right)}$ in (54), we have (14).

C. Proof of Theorem 2

We use the following lemma to prove the result.

Lemma 10. *Suppose that the following holds for a non-negative sequence $\{V(k)\}$ and some non-negative constants $\sigma_i \in [0, 1)$, $0 \leq i \leq \bar{\tau}$ satisfying $\sigma_0 < \sum_{i=0}^{\bar{\tau}} \sigma_i < 1$:*

$$V(k+1) \leq \sum_{i=0}^{\bar{\tau}} \sigma_i V(k-i). \tag{55}$$

Then,

$$V(k) \leq \rho^k V(0), \ \forall k \in \mathbb{N}_0, \tag{56}$$

where

$$\rho = \alpha' b + (1 - \alpha')a. \tag{57}$$

Here, $a = \sum_{i=0}^{\bar{\tau}} \sigma_i$*,* b *can be any scalar in* $[b', 1]$ *where* $b' = a^{\frac{1}{1+(1-\sigma_0)\bar{\tau}}}$*, and*

$$\alpha' = \frac{1}{1 + \frac{b - \sum_{i=0}^{\bar{\tau}} \sigma_i b^{-i}}{\sum_{i=0}^{\bar{\tau}} \sigma_i a^{-i} - a}} \in [0, 1].$$

In addition, $a - \sum_{i=0}^{\bar{\tau}} \sigma_i a^{-i} < 0$ *and* $b - \sum_{i=0}^{\bar{\tau}} \sigma_i b^{-i} \geq 0$*.*

Proof. Define $h(s) = s - \sum_{i=0}^{\bar{\tau}} \sigma_i s^{-i}$. Then,

$$\alpha' = \frac{1}{1 - h(b)/h(a)}. \tag{58}$$

We first show that $h(a) < 0$ and $h(b) \geq 0$. Since $\sigma_0 < \sum_{i=0}^{\bar{\tau}} \sigma_i$ and $a \in (0, 1)$, we have $\sum_{i=1}^{\bar{\tau}} \sigma_i > 0$ and $\sum_{i=1}^{\bar{\tau}} \sigma_i (1 - a^{-i}) < 0$. Therefore,

$$\begin{aligned} h(a) &= a - \sum_{i=0}^{\bar{\tau}} \sigma_i a^{-i} \\ &= \sum_{i=0}^{\bar{\tau}} \sigma_i - \sum_{i=0}^{\bar{\tau}} \sigma_i a^{-i} \\ &= \sum_{i=0}^{\bar{\tau}} \sigma_i (1 - a^{-i}) \\ &= \sum_{i=1}^{\bar{\tau}} \sigma_i (1 - a^{-i}) \\ &< 0. \end{aligned}$$

Moreover, by $b' \in (0, 1)$,

$$\begin{aligned} h(b') &= b' - \sum_{i=0}^{\bar{\tau}} \sigma_i b'^{-i} \\ &= b' - \sigma_0 - \sum_{i=1}^{\bar{\tau}} \sigma_i b'^{-i} \\ &\geq b' - \sigma_0 - \sum_{i=1}^{\bar{\tau}} \sigma_i b'^{-\bar{\tau}} \\ &= b' - \sigma_0 - (a - \sigma_0) b'^{-\bar{\tau}}. \end{aligned} \tag{59}$$

In addition, note that (50) holds for ρ in (46). Then by letting $p = \sigma_0$ and $q = a - \sigma_0$ in (50), we have

$$b'^{-\bar{\tau}} \leq \frac{b' - \sigma_0}{a - \sigma_0},$$

substituting which into (59) gives $h(b') \geq 0$. In addition, $b \geq b'$ and $h(s)$ is an increasing function when $s > 0$. Then we have $h(b) \geq h(b') \geq 0$.

Next, we prove (56) by induction. Clearly, (56) holds for $k = 0$. Suppose that (56) holds for all $k \in [0, k' - 1]$ for some $k' \in \mathbb{N}$. Then, by (55),

$$V(k') \leq \sum_{i=0}^{\bar{\tau}} \sigma_i \rho^{k'-i-1} V(0).$$

To prove (56) with $k = k'$, it suffices to show

$$\sum_{i=0}^{\bar{\tau}} \sigma_i \rho^{k'-i-1} \leq \rho^{k'},$$

or equivalently,

$$\sum_{i=0}^{\bar{\tau}} \sigma_i \rho^{-i} \leq \rho, \tag{60}$$

which is equivalent to $h(\rho) \geq 0$. By $h(a) < 0$, $h(b) \geq 0$, and (58), we have $\alpha' \in (0, 1]$. Then, since h is concave on the non-negative domain, we have

$$\begin{aligned} h(\rho) &= h(\alpha' b + (1-\alpha')a) \\ &\geq \alpha' h(b) + (1-\alpha')h(a) \\ &= 0, \end{aligned}$$

which further yields (60) and also (56) with $k = k'$. Following the induction procedure, we obtain (56) for all $k \in \mathbb{N}_0$. □

Taking expectation on both sides of (43) with respect to the delay $\tau(k)$ and using Assumption 3, we have

$$\mathbb{E}[V(k+1)] \leq \left(\frac{c^2 P_0}{m} + 1 - \frac{1}{m}\right) V(k) + \sum_{i=1}^{\bar{\tau}} \frac{c^2 P_i}{m} V(k-i).$$

Now we use Lemma 10 to derive the convergence of $\mathbb{E}[V(k)]$. Specifically, in Lemma 10 we let $\sigma_0 = 1 - \frac{1}{m} + \frac{c^2 P_0}{m}$, $\sigma_i = \frac{c^2 P_i}{m}$ for all $i \neq 0$, and $b = \rho_a$, so that $a = \rho_c$ and $b' = \rho_c^{\frac{1}{1+\frac{(1-c^2 P_0)\bar{\tau}}{m}}} \leq \rho_c^{\frac{1}{1+\bar{\tau}/m}} = \rho_a$. Then by Lemma 10, we obtain (16). Also by Lemma 10,

$$\phi(\rho_c) < 0, \quad \phi(\rho_a) \geq 0. \tag{61}$$

D. Proof of Proposition 1

For simplicity, we use MEAN$(\cdot)$ and VAR$(\cdot)$ to denote the mean value and variance of probability distributions, respectively.

Letting $n = \bar{\tau} + 1$, $x_i = i - 1$, $\phi(x) = x$, $\alpha_i = P_{i-1}$, and $\beta_i = P'_{i-1}$ in Theorem 1 in (Hadar & Russell, 1969), we have

$$\begin{aligned} \text{MEAN}(\mathcal{P}) &= \sum_{j=0}^{\bar{\tau}} j P_j \\ &\leq \sum_{j=0}^{\bar{\tau}} j P'_j \\ &= \text{MEAN}(\mathcal{P}'), \end{aligned}$$

i.e., $\mathcal{P}$ yields smaller average delay.

If, in addition, MEAN$(\mathcal{P})$ = MEAN$(\mathcal{P}')$. Then by letting $n = \bar{\tau} + 1$, $x_i = i - 1$, $\phi(x) = x^2$, $\alpha_i = P_{i-1}$, and $\beta_i = P'_{i-1}$ in Theorem 1 in (Hadar & Russell, 1969), we have

$$\sum_{j=0}^{\bar{\tau}} j^2 P_j \leq \sum_{j=0}^{\bar{\tau}} j^2 P'_j. \tag{62}$$

In addition, MEAN$(\mathcal{P})$ = MEAN$(\mathcal{P}')$, which, together with (62), yields

$$\begin{aligned} \text{VAR}(\mathcal{P}) &= \sum_{j=0}^{\bar{\tau}} j^2 P_j - (\text{MEAN}(\mathcal{P}))^2 \\ &\leq \sum_{j=0}^{\bar{\tau}} j^2 P'_j - (\text{MEAN}(\mathcal{P}'))^2 \\ &= \text{VAR}(\mathcal{P}'), \end{aligned}$$

i.e., $\mathcal{P}$ yields smaller variance.

E. Proof of Lemma 3

When the delay distribution is specialized to $\mathcal{P}$ and $\mathcal{P}'$, the function ϕ defined in Theorem 2 becomes

$$\text{Distribution } \mathcal{P}: \quad \phi(\rho) := \rho - \rho_c - \frac{c^2}{m}(\sum_{i=0}^{\bar{\tau}} P_i \rho^{-i} - 1),$$
$$\text{Distribution } \mathcal{P}': \quad \phi'(\rho) := \rho - \rho_c - \frac{c^2}{m}(\sum_{i=0}^{\bar{\tau}} P'_i \rho^{-i} - 1).$$

To prove the result, we first show that if $\mathcal{P} \succeq_1 \mathcal{P}'$, then for any $\rho \in (0,1)$,

$$\phi(\rho) \geq \phi'(\rho). \tag{63}$$

By the definitions of $\phi(\rho)$ and $\phi'(\rho)$, (63) holds if

$$\sum_{i=0}^{\bar{\tau}} P_i \rho^{-i} \leq \sum_{i=0}^{\bar{\tau}} P'_i \rho^{-i}. \tag{64}$$

Letting $n = \bar{\tau}+1$, $x_i = i-1$, $\phi(x) = \rho^{-x}$, $\alpha_i = P_{i-1}$, and $\beta_i = P'_{i-1}$ in Theorem 1 in (Hadar & Russell, 1969), we have that if $\mathcal{P} \succeq_1 \mathcal{P}'$, then (64) holds, which further guarantees (63).

By (63) and (61), we have

$$\phi(\rho_a) \overset{(63)}{\geq} \phi'(\rho_a) \overset{(61)}{\geq} 0,$$
$$\phi'(\rho_c) \overset{(63)}{\leq} \phi(\rho_c) \overset{(61)}{<} 0,$$

which implies

$$-\phi(\rho_a)/\phi(\rho_c) \geq -\phi'(\rho_a)/\phi'(\rho_c).$$

Then by (17), $\alpha_{\mathcal{P}} \leq \alpha_{\mathcal{P}'}$. Moreover, $\rho_a \geq \rho_c$, $\rho_{\mathcal{P}} = \alpha_{\mathcal{P}}\rho_a + (1-\alpha_{\mathcal{P}})\rho_c$, and $\rho_{\mathcal{P}'} = \alpha_{\mathcal{P}'}\rho_a + (1-\alpha_{\mathcal{P}'})\rho_c$. Therefore, $\rho_{\mathcal{P}} \leq \rho_{\mathcal{P}'}$. Completes the proof.

F. Proof of Theorem 4

We first define all the notations that will be used in the proof. We define the index sequence $\{\mathcal{I}(t)\}_{t\in\mathbb{N}_0}$ as: $\mathcal{I}(0) = 0$ and for each $t \in \mathbb{N}$,

$$\mathcal{I}(t) = \min\{k' : k - \tau(k) \geq \mathcal{I}(t-1) \text{ for all } k \geq k'\} + 1, \tag{65}$$

and let

$$a(t) := \arg\ \max_{k\in[\mathcal{I}(t),\mathcal{I}(t+1))} V(k)$$

Moreover, we use the same definitions of $V(k)$ and $W(k)$ as in Appendix A:

$$V(k) = \mathbb{E}[\|\mathbf{x}(k) - \mathbf{x}^\star\|^2],$$
$$W(k) = \frac{1-\alpha}{m\alpha}\mathbb{E}[\|(\mathrm{Id} - \mathrm{T})(\mathbf{x}(k))\|^2].$$

To understand the sequence $\mathcal{I}(t)$, note that for each $t \in \mathbb{N}$, by the definition in (65),

$$k - 1 - \tau(k-1) \geq \mathcal{I}(t-1),\ \forall k \geq \mathcal{I}(t). \tag{66}$$

Hence, $\{\mathcal{I}(t)\}$ defines the following *Markov* property for the update (6): *For each $t \in \mathbb{N}$, all the iterates $\mathbf{x}(k)$, $k \geq \mathcal{I}(t)$ are determined by $\mathbf{x}(k)$, $k \in [\mathcal{I}(t-1), \mathcal{I}(t))$ and do not rely on earlier iterates.* Under Assumption 4, the sequence $\{\mathcal{I}(t)\}$ is well defined because given $\mathcal{I}(t)$, there always exists k' such that $k - \tau(k) \geq \mathcal{I}(t)$ for all $k \geq k'$.

The remaining proof includes three steps. Step 1 derives that for any $t \in \mathbb{N}$ and $k \in [\mathcal{I}(t), \mathcal{I}(t+1))$,

$$V(k) \leq \max_{\ell \in [\mathcal{I}(t-1), \mathcal{I}(t))} V(\ell). \tag{67}$$

Base on step 1, step 2 proves

$$V(a(t)) \leq V(a(t-1)) - W(a(t) - 1 - \tau(a(t) - 1)), \tag{68}$$

which is further used in step 3 to show the result.

Step 1: By (39),

$$V(k+1) \leq \max_{k - \tau(k) \leq t \leq k} V(t) - W(k - \tau(k)). \tag{69}$$

With (69), one can prove (67) by induction. To this end, note from (66) that $k - 1 - \tau(k-1) \geq \mathcal{I}(t-1)$ for any $k \in [\mathcal{I}(t), \mathcal{I}(t+1))$. Then, by (69) with $k = \mathcal{I}(t) - 1$, the equation (67) holds naturally for $k = \mathcal{I}(t)$. Suppose that (67) holds for all $k \in [\mathcal{I}(t), k']$ for some $k' \in [\mathcal{I}(t), \mathcal{I}(t+1) - 1)$. Then, by (66) we have $k' - \tau(k') \in [\mathcal{I}(t-1), k']$, which, together with (69) at $k = k'$, yields (67) at $k = k' + 1$. Following this induction procedure we derived (67) for all $k \in [\mathcal{I}(t), \mathcal{I}(t+1))$.

Step 2: By (66),

$$\begin{aligned} &a(t) - 1 - \tau(a(t) - 1) \in [\mathcal{I}(t-1), \mathcal{I}(t+1)), \\ &a(t) - 1 \in [\mathcal{I}(t-1), \mathcal{I}(t+1)) \end{aligned}$$

and therefore

$$[a(t) - 1 - \tau(a(t) - 1), a(t) - 1] \subseteq [\mathcal{I}(t-1), \mathcal{I}(t+1)).$$

Substituting the above equation into (69) at $k = a(t) - 1$ gives

$$V(a(t)) \leq \max_{\ell \in [\mathcal{I}(t-1), \mathcal{I}(t+1))} V(\ell) - W(a(t) - 1 - \tau(a(t) - 1)). \tag{70}$$

In addition, by maximizing the left-hand side of (67) over $k \in [\mathcal{I}(t-1), \mathcal{I}(t+1))$, we have

$$\max_{\ell \in [\mathcal{I}(t-1), \mathcal{I}(t+1))} V(\ell) \leq \max_{\ell \in [\mathcal{I}(t-1), \mathcal{I}(t))} V(\ell) \tag{71}$$

and therefore,

$$\begin{aligned} \max_{\ell \in [\mathcal{I}(t-1), \mathcal{I}(t+1))} V(\ell) &= \max_{\ell \in [\mathcal{I}(t-1), \mathcal{I}(t))} V(\ell) \\ &= V(a(t-1)). \end{aligned} \tag{72}$$

Substituting (72) into (70), we have (68).

Step 3: Adding (68) from $t = 1$ to $t = +\infty$ gives

$$\sum_{t=1}^{\infty} W(a(t) - 1 - \tau(a(t) - 1)) \leq V(a(0)) = V(0),$$

where $V(a(0)) = V(0)$ can be easily derived from (35). Therefore, $\lim_{t \to +\infty} W(a(t) - 1 - \tau(a(t) - 1)) = 0$, which completes the proof.

G. Proof of Theorem 5

We use the same definitions of $V(k)$, $W(k)$, and $\mathcal{I}(t)$ as in Appendix F and let $q = 1 - \frac{1-c^2}{m}$. We first show that for any $t \in \mathbb{N}_0$ and $k \in [\mathcal{I}(t), \mathcal{I}(t+1))$,

$$V(k) \leq q^t V(0). \tag{73}$$

Subsequently, we prove that for any $k \in [\mathcal{I}(t), \mathcal{I}(t+1))$,

$$t \geq \begin{cases} \Theta(k^{1-\beta}), & \beta \in (0,1), \\ \Theta(\ln k) & \beta = 1. \end{cases} \tag{74}$$

Combining the above two equations yields the result.

Proof of (73): Fix $k \in [\mathcal{I}(t)-1, \mathcal{I}(t+1)-1)$. We have by (66) that $k-\tau(k) \in [\mathcal{I}(t-1), \mathcal{I}(t+1)-1)$. Moreover, since any pseudo-contractive operator is also averaged, (71) holds, so that

$$\begin{aligned} V(k) &\leq \max_{\ell \in [\mathcal{I}(t-1), \mathcal{I}(t))} V(\ell), \\ V(k-\tau(k)) &\leq \max_{\ell \in [\mathcal{I}(t-1), \mathcal{I}(t))} V(\ell), \end{aligned}$$

which, together with (43), yields

$$V(k+1) \leq q \max_{\ell \in [\mathcal{I}(t-1), \mathcal{I}(t))} V(\ell).$$

Maximizing the left-hand side of the above equation over $k \in [\mathcal{I}(t)-1, \mathcal{I}(t+1)-1)$, we obtain

$$\begin{aligned} \max_{\ell \in [\mathcal{I}(t), \mathcal{I}(t+1))} V(\ell) &\leq q \max_{\ell \in [\mathcal{I}(t-1), \mathcal{I}(t))} V(\ell) \\ &\leq q^t \max_{\ell \in [\mathcal{I}(0), \mathcal{I}(1))} V(\ell) \\ &= q^t V(0), \end{aligned}$$

where the last step uses $V(k) \leq V(0)\ \forall k \in \mathbb{N}_0$ derived from (69). Therefore, (73) holds.

Proof of (74): We prove (74) by showing that for any $k \in [\mathcal{I}(t), \mathcal{I}(t+1))$,

$$t \geq \begin{cases} \frac{1}{a}(k+\gamma+1)^{1-\beta} - \gamma - 1, & \beta \in (0,1), \\ \frac{\ln(\eta(k+1)/(\gamma+1))}{\ln(1/(1-\eta))} & \beta = 1, \end{cases} \tag{75}$$

where $a = \eta(1-\eta)^{-\beta} + \gamma + 1$. We consider two cases of β separately.

Case 1: $\beta \in (0,1)$. By the definition of $\{\mathcal{I}(t)\}$ in (65), for each $t \in \mathbb{N}_0$,

$$\mathcal{I}(t+1) - 2 - \tau(\mathcal{I}(t+1)-2) \leq \mathcal{I}(t) - 1. \tag{76}$$

Moreover, by Assumption 5,

$$\begin{aligned} \tau(\mathcal{I}(t+1)-2) &\leq \eta(\mathcal{I}(t+1)-2)^{\beta} + \gamma \\ &\leq \eta(\mathcal{I}(t+1)-2) + \gamma, \end{aligned}$$

which, together with (76), yields

$$\mathcal{I}(t+1) - 2 \leq \frac{\mathcal{I}(t)+\gamma-1}{1-\eta}. \tag{77}$$

Then,

$$\begin{aligned} \tau(\mathcal{I}(t+1)-2) &\leq \eta(\mathcal{I}(t+1)-2)^{\beta} + \gamma \\ &\leq \eta\left(\frac{\mathcal{I}(t)+\gamma-1}{1-\eta}\right)^{\beta} + \gamma. \end{aligned}$$

Substituting the above equation into (76) ensures

$$\begin{aligned} \mathcal{I}(t+1) \leq& \mathcal{I}(t) + \eta\left(\frac{\mathcal{I}(t)+\gamma-1}{1-\eta}\right)^{\beta} + \gamma + 1 \\ \leq& \mathcal{I}(t) + (\eta(1-\eta)^{-\beta} + \gamma + 1)(\mathcal{I}(t)+\gamma)^{\beta} \\ =& \mathcal{I}(t) + a(\mathcal{I}(t)+\gamma)^{\beta}, \end{aligned} \tag{78}$$

where the second step uses $\gamma+1 \leq (\gamma+1)\gamma^{\beta} \leq (\gamma+1)(\mathcal{I}(t)+\gamma)^{\beta}$derived from $\gamma \geq 1$. Based on (78), by induction we will show

$$\mathcal{I}(t) \leq (a(t+\gamma))^{\frac{1}{1-\beta}} - \gamma,\ \forall t \in \mathbb{N}_0. \tag{79}$$

When $t=0$, (79) holds because $\gamma \geq 1$, $a \geq 1$, and $\beta \in (0,1)$. Suppose (79) holds for some $t \in \mathbb{N}_0$. Then, by (78) and (79) we have

$$\begin{aligned}\mathcal{I}(t+1) &\leq (a(t+\gamma))^{\frac{1}{1-\beta}} - \gamma + a(a(t+\gamma))^{\frac{\beta}{1-\beta}} \\ &= (a(t+\gamma))^{\frac{\beta}{1-\beta}}(a(t+\gamma+1)) - \gamma \\ &\leq (a(t+\gamma+1))^{\frac{1}{1-\beta}} - \gamma,\end{aligned} \tag{80}$$

i.e., (79) holds for $t+1$. Following the induction procedure we have that (79) holds for all $t \in \mathbb{N}_0$. Then, for any $k \in [\mathcal{I}(t), \mathcal{I}(t+1))$, we have

$$k \leq \mathcal{I}(t+1) - 1 \leq (a(t+\gamma+1))^{\frac{1}{1-\beta}} - \gamma - 1,$$

which further implies (75) with $\beta \in (0,1)$.

Case 2: $\beta = 1$. Adding $\frac{\gamma+1}{\eta}$ to both sides of (77) ensures

$$\mathcal{I}(t+1) + \frac{\gamma+1}{\eta} - 2 \leq \frac{\mathcal{I}(t) + \frac{\gamma+1}{\eta} - 2}{1-\eta},$$

which further yields

$$\begin{aligned}\mathcal{I}(t+1) + \frac{\gamma+1}{\eta} - 2 &\leq (1-\eta)^{-(t+1)}(\mathcal{I}(0) + \frac{\gamma+1}{\eta} - 2) \\ &= (\frac{\gamma+1}{\eta} - 2)(1-\eta)^{-(t+1)} \\ &\leq \frac{\gamma+1}{\eta}(1-\eta)^{-(t+1)}.\end{aligned}$$

Then, for any $k \in [\mathcal{I}(t), \mathcal{I}(t+1))$,

$$k \leq \mathcal{I}(t+1) - 1 \leq \frac{\gamma+1}{\eta}(1-\eta)^{-(t+1)} - 1,$$

which further gives (75) with $\beta = 1$. This completes the proof.

H. Equivalence to ADMM

Running (24)–(25) with $\lambda = 1$ synchronously and indexing the iterate by $k \in \mathbb{N}_0$ ensures

$$z(k) = \frac{1}{m}\sum_{i=1}^{m} x_i(k), \tag{81}$$

$$y_i(k+1) = \mathrm{prox}_{\gamma F_i}(2z(k) - x_i(k)),\ \forall i \in [m], \tag{82}$$

$$x_i(k+1) = x_i(k) + y_i(k+1) - z(k),\ \forall i \in [m], \tag{83}$$

where z_i in (24) is indexed by $z(k)$. ADMM for solving problem (21) takes the following form (Zhang & Kwok, 2014):

$$\hat{y}_i(k+1) = \mathrm{prox}_{F_i/\eta}(\hat{z}(k) - \hat{x}_i(k)/\eta),\ \forall i \in [m], \tag{84}$$

$$\hat{z}(k+1) = \frac{1}{m}\sum_{i=1}^{m}(\hat{y}_i(k+1) + \hat{x}_i(k)/\eta), \tag{85}$$

$$\hat{x}_i(k+1) = \hat{x}_i(k) + \eta(\hat{y}_i(k+1) - \hat{z}(k+1)),\ \forall i \in [m]. \tag{86}$$

It can be verified that if $x_i(k)$, $y_i(k)$, $z_i(k)$ update according to (81)–(83). Then, by letting $\eta = 1/\gamma$, $\hat{x}_i(k) = \eta(x_i(k) - z(k))$, $\hat{y}_i(k) = y_i(k)$, and $\hat{z}(k) = z(k)$, the updates of $\hat{x}_i(k)$, $\hat{y}_i(k)$, $\hat{z}(k)$ follow (84)–(86). One important step of the verification is $\frac{1}{m}\sum_{i=1}^{m}\hat{x}_i(k) = \frac{\eta}{m}\sum_{i=1}^{m}(x_i(k) - z(k)) = \mathbf{0}$, which yields $\hat{z}(k+1) = \frac{1}{m}\sum_{i=1}^{m}\hat{y}_i(k+1)$.

I. Proof of Lemma 7

By Proposition 12.27 in (Bauschke et al., 2011), both $\mathrm{prox}_{\gamma F}$ and $\mathrm{prox}_{\gamma \mathcal{I}_{\mathcal{C}}}$ are $1/2$-averaged (equivalent to firm non-expansiveness in Proposition 12.27 in (Bauschke et al., 2011)). Then, using Proposition 4.21 in (Bauschke et al., 2011), the operator $\mathrm{T}' = \mathrm{prox}_{\gamma F} \circ (2\,\mathrm{prox}_{\gamma \mathcal{I}_{\mathcal{C}}} - \mathrm{Id}) - \mathrm{prox}_{\mathcal{I}_{\mathcal{C}}} + \mathrm{Id}$ is $1/2$-averaged. Hence, $\mathrm{T}' = \frac{R}{2} + \frac{\mathrm{Id}}{2}$ for some non-expansive operator R, so that

$$\mathrm{T} = (1-\lambda)\,\mathrm{Id} + \lambda\,\mathrm{T}' = (1-\lambda/2)\,\mathrm{Id} + (\lambda/2)R.$$

Therefore, T is $\lambda/2$-averaged.

J. Proof of Lemma 8

By the proof of Theorem 25.8 in (Bauschke et al., 2011), the proximal gradient operator $\mathrm{prox}_{\gamma r} \circ (\mathrm{Id} - \gamma \nabla f)$ is $1/(2\theta)$-averaged where $\theta = \frac{\min\{1, L/\gamma\}+1/2}{2}$. Therefore, there exists a non-expansive operator R satisfying $\mathrm{prox}_{\gamma r} \circ (\mathrm{Id} - \gamma \nabla f) = R/(2\theta) + (1 - 1/(2\theta))\,\mathrm{Id}$, so that

$$\begin{aligned}\mathrm{T}' &= \theta\,\mathrm{prox}_{\gamma r} \circ (\mathrm{Id} - \gamma \nabla f) + (1-\theta)\,\mathrm{Id} \\ &= R/2 + \mathrm{Id}\,/2.\end{aligned}$$

Therefore, T' is $1/2$-averaged.

Since T' and $\mathrm{prox}_{\gamma \mathcal{I}_{\mathcal{Z}}}$ are $1/2$-averaged, by using Proposition 4.21 in (Bauschke et al., 2011), the operator $\mathrm{T}' \circ (2\,\mathrm{prox}_{\gamma \mathcal{I}_{\mathcal{Z}}} - \mathrm{Id}) - \mathrm{prox}_{\mathcal{I}_{\mathcal{Z}}} + \mathrm{Id}$ is $1/2$-averaged. Then, $\mathrm{T}' \circ (2\,\mathrm{prox}_{\gamma \mathcal{I}_{\mathcal{Z}}} - \mathrm{Id}) - \mathrm{prox}_{\mathcal{I}_{\mathcal{Z}}} + \mathrm{Id} = \frac{R}{2} + \frac{\mathrm{Id}}{2}$ for some non-expansive operator R, so that

$$\begin{aligned}\mathrm{T} &= (1-\lambda)\,\mathrm{Id} + \lambda(\mathrm{T}' \circ (2\,\mathrm{prox}_{\gamma \mathcal{I}_{\mathcal{Z}}} - \mathrm{Id}) - \mathrm{prox}_{\mathcal{I}_{\mathcal{Z}}} + \mathrm{Id}) \\ &= (1-\lambda/2)\,\mathrm{Id} + (\lambda/2)R.\end{aligned}$$

Therefore, T is $\lambda/2$-averaged.

K. Delay distribution

To validate the phenomenon that most delays may be much smaller than the maximum delay, we plot the delay distribution generated by the experiments (theoretical parameter) in Section 4. Specifically, for both Lasso (Figure 2) and Logistic regression (Figure 3), we plot the delay distribution generated by all four asynchronous algorithms.

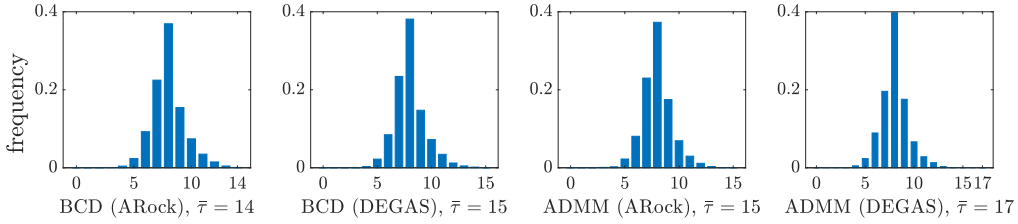
Observe from Figure 5 that most delays are much smaller than the maximum delay. For example, in Figure 5 (a) BCD (ARock), the maximum delay is 14 and over 86% delays are smaller than or equal to 9, and in Figure 5 (b) ADMM (ARock), the maximum delay is 28 while over 97% delays are smaller than or equal to 11.

L. Comparison in terms of running time

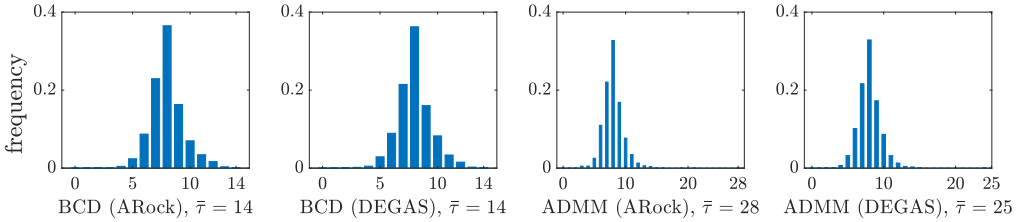
We conducted a comparison between DEGAS, ARock, and their common synchronous algorithms based on their running time. We consider two scenarios: no straggler and with straggler. In the first setting, we use the same experiment setting as in Section 4, where each worker is a core in a 10-core machine and all the workers are homogeneous. In the second setting, we choose one worker as the straggler and let it sleep for twice its local computation time at each iteration. The "sleep" scheme for setting a straggler is standard in the literature (Lian et al., 2018; Luo et al., 2020). However, existing works usually choose a random straggler at each iteration, while we consider the more practical setting where the straggler is fixed. We use the theoretical step-size setting in Section 4.

The experiment results are presented in Figures 6–7. Observe from Figures 6–7 that DEGAS is slower than the synchronous methods if there is no straggler, and is faster in the case of including straggler. This phenomenon is reasonable. First, without a straggler, the numbers of T_i computed by the synchronous and asynchronous methods are very close according to our observation, while the performance of asynchronous methods is degraded by the information delay. Second, when a straggler is present, the average per-iteration time consumption of the synchronous methods significantly increases, while that of the asynchronous methods only increases slightly.

Figure 5. Delay distribution (x-axis represents delay)

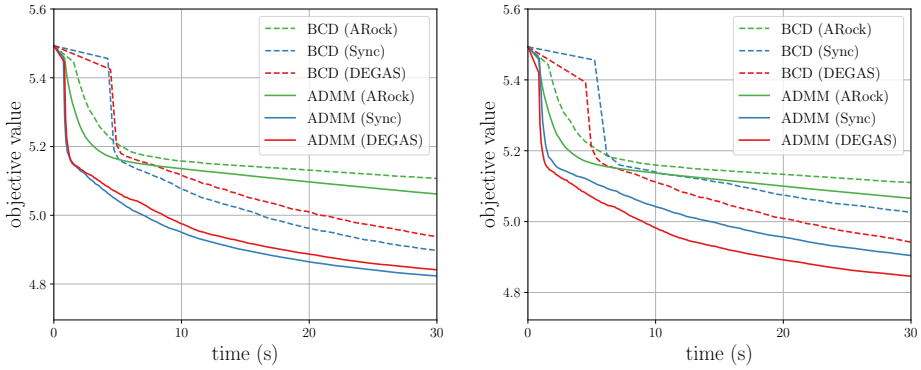


(a) Lasso



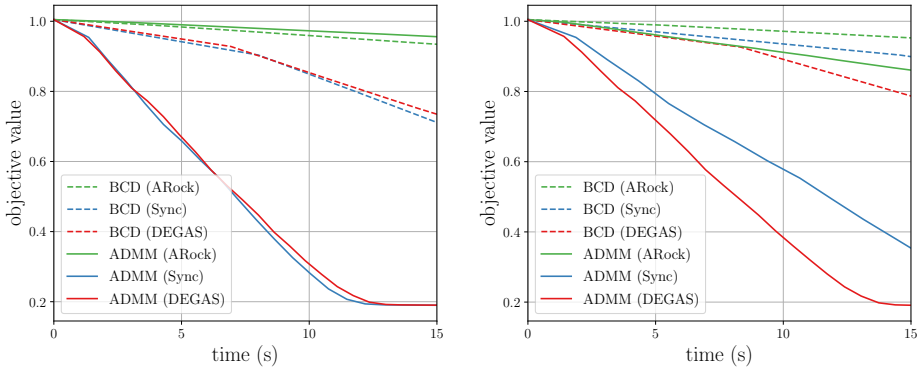
(b) Logistic

Figure 6. Convergence for solving Lasso: no straggler v.s. with straggler



(a) no straggler (b) with straggler

Figure 7. Convergence for solving Logistic regression: no straggler v.s. with straggler



(a) no straggler (b) with straggler