Asynchronous Gradient-Push

Mahmoud Assran and Michael Rabbat

***Abstract*—We consider a multi-agent framework for distributed optimization where each agent has access to a local smooth strongly convex function, and the collective goal is to achieve consensus on the parameters that minimize the sum of the agents' local functions. We propose an algorithm wherein each agent operates asynchronously and independently of the other agents. When the local functions are strongly-convex with Lipschitz-continuous gradients, we show that the iterates at each agent converge to a neighborhood of the global minimum, where the neighborhood size depends on the degree of asynchrony in the multi-agent network. When the agents work at the same rate, convergence to the global minimizer is achieved. Numerical experiments demonstrate that Asynchronous Gradient-Push can minimize the global objective faster than state-of-the-art synchronous first-order methods, is more robust to failing or stalling agents, and scales better with the network size.**

I. Introduction

We propose and analyze an asynchronous distributed algorithm to solve the optimization problem

$$\text{minimize}_{x\in\mathbb{R}^d} \quad F(x) := \sum_{i=1}^n f_i(x) \tag{1}$$

where each $f_i : \mathbb{R}^d \to \mathbb{R}$ is smooth and strongly convex. We focus on the multi-agent setting, in which there are n agents and information about the function f_i is only available at the i^{th} agent. Specifically, only the i^{th} agent can evaluate f_i and gradients of f_i. Consequently, the agents must cooperate to find a minimizer of F.

Many multi-agent optimization algorithms have been proposed, motivated by a variety of applications including distributed sensing systems, the internet of things, the smart grid, multi-robot systems, and large-scale machine learning. In general, there have been significant advances in the development of distributed methods with theoretical convergence guarantees in a variety of challenging scenarios such as time-varying and directed graphs (see [1] for a recent survey). However, the vast majority of this literature has focused on *synchronous* methods, where all agents perform updates at the same rate.

This paper studies asynchronous distributed algorithms for multi-agent optimization. Our interest in this setting comes from applications of multi-agent methods to solve large-scale optimization problems arising in the context of machine learning, where each agent may be running on a different server and the agents communicate over a wired network. Hence, agents may receive multiple messages from their neighbours at any given time instant, and may perform a drastically different number of gradient steps over any time interval. In distributed computing systems, communication delays may be unpredictable; communication links may be unreliable; and each processor may be shared for other tasks while at the same time cooperating with other processors in the context of some computational task [2]. High performance computing clusters fit this model of distributed computing quite nicely [3], especially since node and link failures may be expected in such systems [4]–[6]. When a synchronous algorithm is run in such a setting, the rate of progress of the entire system is hampered by the slowest node or communication link; asynchronous algorithms are largely immune to such issues [2], [7]–[14].

The authors are with Facebook AI Research, Montréal, Québec, Canada, and the Department of Electrical and Computer Engineering, McGill University, Montréal, Québec, Canada. Email: {massran, mikerabbat}@fb.com.

A. Asynchronous Gradient-Push

Practical implementations of multi-agent communication—using the Message Passing Interface (MPI) [15] or other message passing standards—often have the notion of a *send-buffer* and a *receive-buffer*. A send-buffer is a data structure containing the messages sent by an agent, but not yet physically transmitted by the underlying communication system. A receive-buffer is a data structure containing the messages received by an agent, but not yet processed by the application.

Using this notion of send- and receive-buffers, the individual-agent pseudocode for running the asynchronous gradient-push method is shown in Algorithm 1. The method repeats a two-step procedure consisting of **Local Computation** followed by **Asynchronous Gossip**. During the **Local Computation** phase, agents update their estimate of the minimizer by performing a local (sub)gradient-descent step. During the **Asynchronous Gossip** phase, agents copy all outgoing messages into their local send-buffer and subsequently process (sum) all messages received (buffered) in their local receive-buffer while the agent was busy performing the preceding **Local Computation**. The underlying communication system begins transmitting the messages in the local send-buffer once they are copied there; thereby freeing the agent to proceed to the next step of the algorithm without waiting for the messages to reach their destination.

Fig. 1a illustrates the agent update procedure in the synchronous case: agents must wait for all network communications to be completed before moving-on to the next iteration, and, as a result, some agents may experience idling periods. Fig. 1b illustrates the agent update procedure in the asynchronous case: at the beginning of each local iteration, agents make use of their message buffers by copying all outgoing messages into their local send-buffers, and by retrieving all messages from their local receive-buffers. The underlying communication systems subsequently transmit the messages in the send-buffers while the agents proceed with their computations.

B. Related Work

Most multi-agent optimization methods are built on distributed averaging algorithms [16]. For synchronous methods

Algorithm 1 Asynchronous Gradient-Push (Pseudocode) for agent v_i

Initialize $x_i \in \mathbb{R}^d$ ▷ Push-sum numerator
Initialize $y_i \leftarrow 1$ ▷ Push-sum weight
Initialize $\alpha_i > 0$ ▷ Step-size
$N_i^{\text{out}} \leftarrow$ number of out-neighbours of v_i
while stopping criterion not satisfied **do**
  Begin: Local Computation
  $z_i \leftarrow x_i / y_i$ ▷ De-biased consensus estimate
  $x_i \leftarrow x_i - \alpha_i \nabla f_i(z_i)$
  Update step-size α_i
  Begin: Asynchronous Gossip
  Copy message $(x_i/N_i^{\text{out}}, y_i/N_i^{\text{out}})$ to local send-buffer
  $x_i \leftarrow x_i/N_i^{\text{out}} + \sum_{(x',y') \in \text{receive-buffer}} x'$
  $y_i \leftarrow y_i/N_i^{\text{out}} + \sum_{(x',y') \in \text{receive-buffer}} y'$
end while

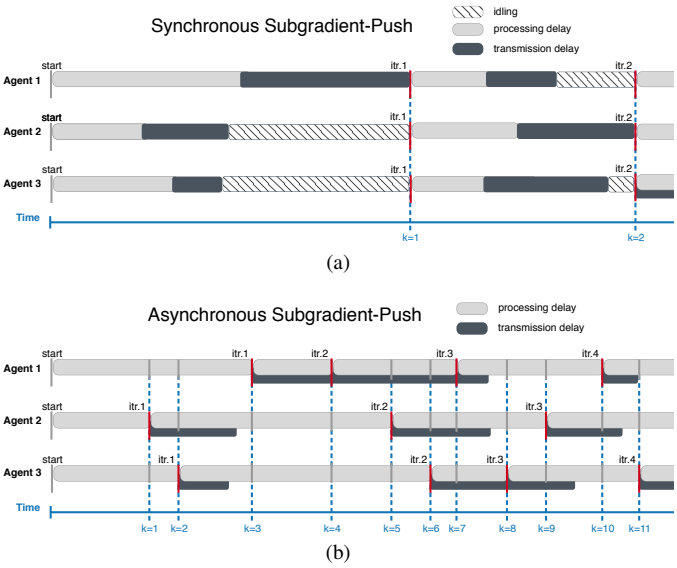


Fig. 1. Example of agent updates in synchronous and asynchronous Gradient-Push implementations. Processing delays correspond to the time required to perform a local iteration. Transmission delays correspond to the time required for all outgoing message to arrive at their destination buffers. Even though a message arrives at a destination agent's receive-buffer after some real (non-integer valued) delay, that message is only processed when the destination agents performs its next update.

operating over static, undirected networks, it is possible to use doubly stochastic averaging matrices. However, it turns out that averaging protocols which rely on doubly stochastic matrices may be undesirable for a variety of reasons [4]. The Push-Sum approach to distributed averaging, introduced in [5], eliminates the need for doubly stochastic consensus matrices. The seminal work on Push-Sum [5] analyzed convergence for complete network topologies (all pairs of agents may communicate directly). The analysis was extended in [17] for general connected graphs. Further work has provided convergence guarantees in the face of the other practical issues, such as communication delays and dropped messages [18], [19]. In general, Push-Sum is attractive for implementations because it can easily handle directed communication topologies, and thus avoids incidents of deadlock that may occur in practice when using undirected communication topologies [4].

Multi-Agent Optimization with Column Stochastic Consensus Matrices: The first multi-agent optimization algorithm using Push-Sum for distributed averaging was proposed in [20]. Nedić and Olshevsky [21] continue this line of work by introducing and analyzing the Subgradient-Push method; the analysis in [21] focuses on minimizing (weakly) convex, Lipschitz functions, for which diminishing step-sizes are required to obtain convergence. Xi and Khan [22] propose DEXTRA and Zeng and Yin [23] propose Extra-Push, both of which use the Push-Sum protocol in conjunction with gradient tracking to achieve geometric convergence for smooth, strongly convex objectives over directed graphs. Nedić, Olshevsky, and Shi [24] propose the Push-DIGing algorithm, which achieves a geometric convergence rate over directed and time-varying communication graphs. Push-DIGing and DEXTRA/Extra-Push are considered to be state-of-the-art synchronous methods, and the Subgradient-Push algorithm is a multi-agent analog of classical gradient descent. It should be noted that all of these algorithms are *synchronous* in nature.

Asynchronous Multi-Agent Optimization: The seminal work on asynchronous distributed optimization of Tsitsiklis *et al.* [25] considers the case where each agent holds one component of the optimization variable (or the entire optimization variable), and can locally evaluate a descent direction with respect to the global objective. Convergence is proved for a distributed gradient algorithm in that setting. However that setting is inherently different from the proposed problem formulation in which each agent does not necessarily have access to the global objective. Li and Basar [26] study distributed asynchronous algorithms and prove convergence and asymptotic agreement in a stochastic setting, but assume a similar computation model to that of Tsitsiklis *et al.* [25] in which each agent updates a portion of the parameter vector using an operator which produces contractions with respect to the global objective.

Recently, several asynchronous multi-agent optimization methods have been proposed, such as: [14], which requires doubly-stochastic consensus over undirected graphs; [8], [27], which require push-pull consensus over undirected graphs; and [28], which assumes a model of asynchrony in which agents become activated according to a Poisson point process, and an active agent finishes its update before the next agent becomes activated. In general, many of the asynchronous multi-agent optimization algorithms in the literature make restrictive assumptions regarding the nature of the agent updates (*e.g.*, sparse Poisson point process [28], randomized single activation [29], [30], randomized multi-activation [31]–[34]).

C. Contributions and Paper Organization

We study an asynchronous implementation of the Subgradient-Push algorithm. Since we focus on problems with continuously differentiable objectives, we refer to the method as *asynchronous Gradient-Push* (AGP). This paper draws motivation from our previous work [9] in which we

empirically studied AGP and observed that it converges faster than state-of-the-art synchronous multi-agent algorithms. In this paper we provide theoretical convergence guarantees: when the local objective functions are strongly convex with Lipschitz-continuous gradients, we show that the iterates at each agent achieve consensus and converge to a neighborhood of the global minimum, where the size of the neighborhood depends on the degree of asynchrony. We consider a model of asynchrony which allows for heterogenous, bounded computation delays and communication delays. When the agents work at the same rate, convergence to the global minimizer is achieved. Moreover, if agents have knowledge of one another's potentially time-varying update rates, then they can set their step-sizes to achieve convergence to the global minimizer. In general, we relate the asymptotic worst-case error to the degree of asynchrony, as quantified by a bound on the delay. Agents do not need to know the delay bounds to execute the algorithm; the bounds only appear in the analysis.

Our analysis is based on several novel aspects: whereas previous work has used graph augmentation to model communication delays in consensus algorithms, here we augment with virtual nodes to model the effects of both communication *and* computation delays on message passing in optimization algorithms. Combining the graph augmentation with a (possibly time-varying) binary-valued activation function that is unique to each agent and directly multiplies its step-size, we are able to model the effect of heterogeneous update rates on the optimization procedure. In contrast to previous work that makes additional assumptions on the agents' update rates, our problem formulation only assumes that the time-interval between an agents' consecutive activations is bounded. Specifically, this formulation readily allows us to characterize the limit point as a *deterministic function* of the agents' update rates, and to bound the rate of convergence when running AGP with constant or diminishing step-sizes. Since synchronous gradient-push is a special case of AGP (with zero communication delay and unit computation delays), we obtain the first theoretical convergence guarantees for gradient-push with constant step-size.

We also develop peripheral results concerning an asynchronous version of the Push-Sum protocol used for consensus averaging that may be of independent interest. In particular, we show that agents running the Push-Sum protocol asynchronously converge to the average of the network geometrically fast, even in the presence of exogenous perturbations at each agent, where the constant of geometric convergence depends on the consensus-matrices' degree of ergodicity [35] and a measure of asynchrony in the network.

In Sec. II we describe the model of asynchrony considered in this paper. In Sec. III we expound the Asynchronous Perturbed Push-Sum consensus averaging protocol and give the associated convergence results. In Sec. IV we formally describe the AGP optimization algorithm and present our main convergence results for both the constant and diminishing step-size cases. Sec. V is devoted to the proof of the main results, and in Sec. VI we report numerical experiments on a high performance computing cluster. Finally, in Sec. VII, we conclude and discuss extensions for future work.

II. System Model

A. Communication

The multi-agent communication topology is represented by a directed graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$, where

$$\mathcal{V} := \{v_i \mid i = 1, \dots, n\},$$
$$\mathcal{E} := \{(v_j \leftarrow v_i) \mid v_i \text{ can send messages to } v_j\},$$

are the set of agents and edges respectively. We refer to $\mathcal{G}(\mathcal{V}, \mathcal{E})$ as the *reference graph* for reasons that will become apparent when we augment the graph with virtual agents. Let

$$N_j^{\text{in}} := \mathbf{card}\left(\{v_i \mid (v_j \leftarrow v_i) \in \mathcal{E}\}\right)$$
$$N_j^{\text{out}} := \mathbf{card}\left(\{v_i \mid (v_i \leftarrow v_j) \in \mathcal{E}\}\right)$$

denote the cardinality of the *in-neighbor* set and *out-neighbor* set of agent v_j, respectively; we adopt the convention that $(v_i \leftarrow v_i) \in \mathcal{E}$ for all i, *i.e.*, every agent can send messages to itself.

B. Discrete event sequence

Without any loss of generality we can describe and analyze asynchronous algorithms as discrete sequences since all events of interest, such as message transmissions/receptions and local variable updates, may be indexed by a discrete-time variable [25]. We adopt notation and terminology for analyzing asynchronous algorithms similar to that developed in [25]. Let $t[0] \in \mathbb{R}_+$ denote the time at which the agents begin optimization. We assume that there is a set of times $T = \{t[1], t[2], t[3], \dots, \}$ at which one or more agents become *activated*; i.e., completes a **Local Computation** and begins **Asynchronous Gossip**. Let $T_i \subseteq T$ denote the subset of times at which agent v_i in particular becomes activated. Let $\mathcal{A}[k] := \{v_i \mid t[k] \in T_i\}$ denote the *activation set* at time-index $k \in \mathbb{N}$, which is the set of agents that are activated at time $t[k]$. For convenience, we also define the functions $\pi_i(k) := \max\{k' \in \mathbb{N} \mid k' < k,\ v_i \in \mathcal{A}[k']\}$ for all i, which return the most recent time-index — up to, but not including, time-index k — when agent v_i was in the activation set.[1]

C. Delays

Recall that $t[k] \in T_i$ denotes a time at which agent v_i becomes *activated*: it completes a **Local Computation** (i.e., performs an update) and begins **Asynchronous Gossip** (i.e., sends a message to its neighbours by copying the outgoing message into its local send-buffer). For analysis purposes, messages are sent with an *effective delay* such that they arrive right when the agent is ready to process the messages. That is, a message that is sent at time $t[k]$ and processed by the receiving agent at time $t[k']$, where $k' > k$, is treated as having experienced a time delay $t[k'] - t[k]$ for the purpose of analysis, or equivalently a time-index delay $k' - k$, even if the message actually arrives before $t[k']$ and waits in the receive-buffer.

Let $\tau_i^{\text{proc}}[k] := k - \pi_i(k)$ (defined for all k such that $t[k] \in T_i$) denote the time-index processing delay experienced by agent v_i at time $t[k]$. In words, if agent v_i performs an update

[1] To handle the corner-case at $k = 1$, we let $\pi_i(1)$ equal 0 for all i.

at some time $t[k]$, then it performed its last update at time $t[k - \tau_i^{\text{proc}}[k]]$. We assume that there exists a constant $\overline{\tau}^{\text{proc}} < \infty$ independent of i and k such that $1 \leq \tau_i^{\text{proc}}[k] \leq \overline{\tau}^{\text{proc}}$.

Similarly, let $\tau_{ji}^{\text{msg}}[k]$ (defined for all k such that $t[k] \in T_j$) denote the time-index message delay experienced by a message sent from agent v_i to agent v_j at time $t[k]$. In words, if agent v_i sends a message to agent v_j at time $t[k]$, then agent v_j will begin processing that message at time $t[k + \tau_{ji}^{\text{msg}}[k]]$. We assume that there exists a constant $\overline{\tau}^{\text{msg}} < \infty$ independent of i, j, and k, such that $\tau_{ji}^{\text{msg}}[k] \leq \overline{\tau}^{\text{msg}}$. In addition, we use the convention that $\tau_{ii}^{\text{msg}}[k] = 0$ for all i and $k \in \mathbb{N}$, meaning that agents always have immediate access to their most recent local variables. Thus $0 \leq \tau_{ji}^{\text{msg}}[k] \leq \overline{\tau}^{\text{msg}}$.

Since all agents enter the activation set (*i.e.*, complete an update and initiate a message transmission to all their out-neighbors) at least once every $\overline{\tau}^{\text{proc}} - 1$ time-indices, and because all messages are processed within at most $\overline{\tau}^{\text{msg}}$ time-indices from when they are sent, it follows that each agent is guaranteed to process at least one message from each of its in-neighbors every $\overline{\tau} := \overline{\tau}^{\text{msg}} + \overline{\tau}^{\text{proc}} - 1$ time-indices.

D. Augmented Graph

To analyze the AGP optimization algorithm we augment the reference graph by adding $\overline{\tau}^{\text{msg}}$ virtual agents for each non-virtual agent. Similar graph augmentations have been used in [18], [19] for synchronous averaging with transmission delays. One novel aspect of the augmentation described here is the use of virtual agents to model the effects of computation delays on message passing. To state the procedure concisely: for each non-virtual agent, v_j, we add $\overline{\tau}^{\text{msg}}$ virtual agents, $v_j^{(1)}, v_j^{(2)}, \ldots, v_j^{(\overline{\tau}^{\text{msg}})}$, where each $v_j^{(r)}$ contains the messages to be received by agent v_j in r time-indices. As an aside, we may interchangeably refer to the non-virtual agents, v_j, as $v_j^{(0)}$ for the purpose of notational consistency. The virtual agents associated with agent v_j are daisy-chained together with edges $(v_j^{(r-1)} \leftarrow v_j^{(r)})$, such that at each time-index k, and for all $r = 1, 2, \ldots, \overline{\tau}^{\text{msg}}$, agent $v_j^{(r)}$ forwards its summed messages to agent $v_j^{(r-1)}$. In addition, for each edge $(v_j^{(0)} \leftarrow v_i^{(0)})$ in the reference graph (where $j \neq i$), we add the edges $(v_j^{(r)} \leftarrow v_i^{(0)})$ in the augmented graph. This augmented model simplifies the subsequent analysis by enabling agent v_i to send a message to agent $v_j^{(r)}$ with delay zero, rather than send a message to agent v_j with delay r.[2] See Fig. 2 for an example.

To adapt the augmented graph model for optimization we formulate the equivalent optimization problem

$$\text{minimize} \;\; \overline{F}(x) := \sum_{r=0}^{\overline{\tau}^{\text{msg}}} \sum_{i=1}^{n} f_i^{(r)}(x), \tag{2}$$

where

$$f_i^{(r)}(x) = \begin{cases} f_i(x) & \text{if } r = 0, \\ 0 & \text{otherwise.} \end{cases}$$

[2]It is worth pointing out that we have not changed our definitions for the edge and vertex sets $\mathcal{E}$ and $\mathcal{V}$ respectively; they are still solely defined in-terms of the non-virtual agents.

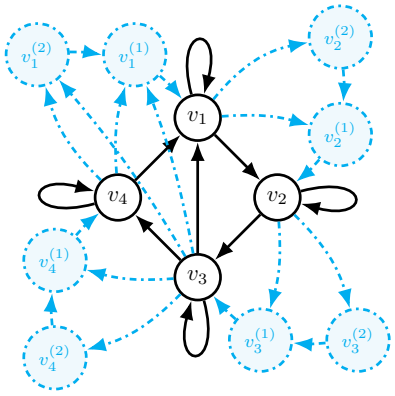


Fig. 2. Sample augmented graph of a 4-agent reference network with a maximum time-index message transmission delay of $\overline{\tau}^{\text{msg}} = 2$ time-indices. Solid lines correspond to non-virtual agents and edges. Dashed lines correspond to virtual agents and edges.

In words, each of the non-virtual agents, $v_i^{(0)}$, maintains its original objective function $f_i(\cdot)$, and all the virtual agents are simply given the zero objective. Clearly $\overline{F}(x)$ defined in (2) is equal to $F(x)$ defined in (1). We denote the state of a variable x at time $t[k]$ with an augmented state matrix $\boldsymbol{x}[k] \in \mathbb{R}^{n(\overline{\tau}^{\text{msg}}+1) \times d}$

$$\boldsymbol{x}[k] := \begin{bmatrix} \boldsymbol{x}^{(0)}[k] \\ \boldsymbol{x}^{(1)}[k] \\ \vdots \\ \boldsymbol{x}^{(\overline{\tau}^{\text{msg}})}[k] \end{bmatrix}, \tag{3}$$

where each $\boldsymbol{x}^{(r)}[k] \in \mathbb{R}^{n \times d}$ is a block matrix that holds the copy of the variable x at all the delay-r agents in the augmented graph at time-index k.[3] More specifically, $x_i^{(r)}[k] \in \mathbb{R}^d$, the i^{th} row of $\boldsymbol{x}^{(r)}[k]$, is the local copy of the variable x held locally at agent $v_i^{(r)}$ at time-index k; below we generalize this notation for other variables as well.

For ease of exposition, we assume that the reference-graph is static and strongly-connected. The strongly-connected property of the directed graph is necessary to ensure that all agents are capable of influencing each other's values, and in Sec. VII we describe how one can extend our analysis to account for time-varying directed communication-topologies.

III. Asynchronous Perturbed Push-Sum

Consensus-averaging is a fundamental building block of the proposed AGP algorithm. In this subsection we consider an asynchronous version of the synchronous Perturbed Push-Sum Protocol [21]. If we omit the gradient update in line 8 of Algorithm 1, then we recover the pseudocode for an asynchronous formulation of the Push-Sum consensus averaging protocol. Alternatively, if we replace the gradient term in line 8 of Algorithm 1 with a general perturbation term, then we recover an asynchronous formulation of the Perturbed Push-Sum consensus averaging protocol.

[3]In keeping with this notation, the block matrix $\boldsymbol{x}^{(0)}[k]$ corresponds to the non-virtual agents in the network.

Algorithm 2 Asynchronous Perturbed Push-Sum Averaging

for $k = 0, 1, 2, \ldots$ **to** termination **do**

$$\boldsymbol{x}[k+1] = \overline{\boldsymbol{P}}[k]\left(\boldsymbol{x}[k] + \boldsymbol{\eta}[k]\right) \tag{6}$$

$$y[k+1] = \overline{\boldsymbol{P}}[k] y[k] \tag{7}$$

$$\boldsymbol{z}[k+1] = \text{diag}(y[k+1])^{-1}\boldsymbol{x}[k+1] \tag{8}$$

A. Formulation of Asynchronous (Perturbed) Push-Sum

We describe the Asynchronous Perturbed Push-Sum algorithm in matrix form (which will facilitate analysis below) by stacking all of the agents' parameters at every update time into a parameter matrix using a similar notation to that in (3). The entire **Asynchronous Gossip** procedure can then be represented by multiplying the parameter-matrix by a so-called *consensus-matrix* that conforms to the graph structure of the communication topology. The consensus matrices $\overline{\boldsymbol{P}}[k] \in \mathbb{R}^{n(\overline{\tau}^{\text{msg}}+1)\times n(\overline{\tau}^{\text{msg}}+1)}$ for the augmented state model are defined as

$$\overline{\boldsymbol{P}}[k] := \begin{bmatrix} \widetilde{\boldsymbol{P}}_0[k] & \boldsymbol{I}_{n\times n} & \boldsymbol{0} & \cdots & \boldsymbol{0} \\ \widetilde{\boldsymbol{P}}_1[k] & \boldsymbol{0} & \boldsymbol{I}_{n\times n} & \cdots & \boldsymbol{0} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \widetilde{\boldsymbol{P}}_{\overline{\tau}-1}[k] & \boldsymbol{0} & \boldsymbol{0} & \cdots & \boldsymbol{I}_{n\times n} \\ \widetilde{\boldsymbol{P}}_{\overline{\tau}^{\text{msg}}}[k] & \boldsymbol{0} & \boldsymbol{0} & \cdots & \boldsymbol{0} \end{bmatrix}, \tag{4}$$

where each $\widetilde{\boldsymbol{P}}_r[k] \in \mathbb{R}^{n\times n}$ is a block matrix defined as

$$\left[\widetilde{\boldsymbol{P}}_r[k]\right]_{ji} := \begin{cases} \frac{1}{N_i^{\text{out}}}, & v_i \in \mathcal{A}[k],\ (j,i) \in \mathcal{E},\ \text{and } \tau_{ji}^{\text{msg}}[k] = r, \\ 1, & v_i \notin \mathcal{A}[k],\ r = 0,\ j = i, \\ 0, & \text{otherwise.} \end{cases} \tag{5}$$

In words, when a non-virtual agent is in the activation set, it sends a message to each of its out-neighbours in the reference graph with some arbitrary, but bounded, delay r. When a non-virtual agent is not in the activation set, it keeps its value and does not gossip. Furthermore, since we have chosen a convention in which messages between agents are sent with some effective message delay, $\tau_{ji}^{\text{msg}}[k]$, it follows that non-virtual agents do not receive any new messages while outside the activation set. Virtual agents, on the other hand, simply forward all of their messages to the next agent in the delay chain at all time-indices k, and so there is no notion of virtual agents belonging to (or not belonging to) the activation set. The activation set is exclusively a construct for the non-virtual agents. Observe that, by definition, the matrices $\overline{\boldsymbol{P}}[k]$ are column stochastic at all time-indices k.

To analyze the Asynchronous Perturbed Push-Sum averaging algorithm from a global perspective, we use the matrix-based formulation provided in Algorithm 2, where $\boldsymbol{\eta}[k] \in \mathbb{R}^{n(\overline{\tau}^{\text{msg}}+1)\times d}$ is a perturbation term, and the matrices $\overline{\boldsymbol{P}}[k]$ are as defined in (4) for the augmented state, and $\boldsymbol{x}[k]$, $y[k]$, and $\boldsymbol{z}[k]$ are also defined with respect to the augmented state. At all time-indices k, each agent $v_i^{(r)}$ locally maintains the variables $x_i^{(r)}[k], z_i^{(r)}[k], \in \mathbb{R}^d$, and $y_i^{(r)}[k] \in \mathbb{R}$. The non-virtual agent initializations are $x_i^{(0)}[0] \in \mathbb{R}^d$, and $y_i^{(0)}[0] = 1$. The virtual agent initializations are $x_i^{(r)}[0] = \boldsymbol{0}$, and $y_i^{(r)}[0] = 0$ (for all $r \neq 0$).[4] This matrix-based formulation describes how the agents' values evolve at those times $t[k+1] \in T = \{t[1], t[2], t[3], \ldots,\}$ when one or more agents complete an update, which in this case consists of summing received messages. The time-varying consensus-matrices $\overline{\boldsymbol{P}}[\cdot]$ capture the asynchronous delay-prone communication dynamics.

B. Main Results for Asynchronous (Perturbed) Push-Sum

In this subsection we present the main convergence results for the Asynchronous (Perturbed) Push-Sum consensus averaging protocol. We briefly describe some notation in order to state the main results. Let $N_{max}^{\text{out}} := \max_{1\leq j\leq n} N_j^{\text{out}}$ represent the maximum number of out-neighbors associated to any non-virtual agent. Let $\overline{x}[k] := \boldsymbol{1}^\top \boldsymbol{x}[k]/n$ be the mutual time-wise average of the variable x at time-index k. Let the scalar ψ represent the number of possible types (zero/non-zero structures) that an $n(\overline{\tau}^{\text{msg}}+1) \times n(\overline{\tau}^{\text{msg}}+1)$ stochastic, indecomposable, and aperiodic (SIA) matrix can take (hence $\psi < 2^{(n(\overline{\tau}^{\text{msg}}+1))^2}$).[5] Let the scalar $\lambda > 0$ represent the maximum Hajnal Coefficient of Ergodicity [35] taken over the product of all possible $(\overline{\tau}+1)(\psi+1)$ consecutive consensus-matrix products:

$$\lambda := \max_{\boldsymbol{A}} \left(1 - \min_{j_1, j_2} \sum_i \min\left\{[\boldsymbol{A}]_{i,j_1}, [\boldsymbol{A}]_{i,j_2}\right\}\right),$$

such that

$$\boldsymbol{A} \in \left\{\overline{\boldsymbol{P}}[k + (\overline{\tau}+1)(\psi+1)] \cdots \overline{\boldsymbol{P}}[k+2]\overline{\boldsymbol{P}}[k+1] \;\middle|\; k \geq 0\right\},$$

where $\overline{\tau} := \overline{\tau}^{\text{msg}} + \overline{\tau}^{\text{proc}} - 1$. We prove that λ is strictly less than 1 and guaranteed to exist. Let δ_{min} represent a lower bound on the entries in the first n-rows of the product of $n(\overline{\tau}+1)$ or more consecutive consensus-matrices (rows only corresponding to the non-virtual agents):

$$\delta_{min} := \min_{i,j,k,\ell} \left[\overline{\boldsymbol{P}}[k+\ell] \cdots \overline{\boldsymbol{P}}[k+2]\overline{\boldsymbol{P}}[k+1]\right]_{i,j},$$

where the min is taken over all $i = 1, 2, \ldots, n$, $j = 1, 2, \ldots, n(\overline{\tau}^{\text{proc}}+1)$, $k \geq 0$, and $\ell \geq n(\overline{\tau}^{\text{proc}}+1)$.

Assumption 1 (Communicability)**.** *All agents influence each other's values sufficiently often, in particular:*

1) *The reference graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$ is static and strongly connected.*
2) *The communication and computation delays are bounded: $\overline{\tau}^{msg} < \infty$ and $\overline{\tau}^{proc} < \infty$.*

[4] Note, given the initializations, the virtual agents could potentially have $z_i^{(r)}[k+1] = 0/0$ (division by zero) in update equation (11), but this is a non-issue since $z_i^{(r)}$ (for all $r \neq 0$) is never used.

[5] See [36] for a definition of SIA matrices.

Theorem 1 (Convergence Rate of Asynchronous Perturbed Push-Sum Averaging)**.** *Suppose that Assumption 1 is satisfied. Then it holds for all $i = 1, 2, \dots, n$, and $k \geq 0$, that*

$$\left\| z_i^{(0)}[k] - \overline{x}[k] \right\|_1 \leq C q^k \left\| x_i^{(0)}[0] \right\|_1 + C \sum_{s=0}^{k} q^{k-s} \left\| \eta_i[s] \right\|_1,$$

where $q \in (0,1)$ and $C > 0$ are given by

$$q = \lambda^{\frac{1}{(\psi+1)(\overline{\tau}+1)}}, \quad \text{and} \quad C < \frac{2}{\lambda^{\frac{\psi+2}{\psi+1}} \delta_{min}} \approx \frac{2}{\lambda \delta_{min}},$$

and $\delta_{min} = \left(\frac{1}{N_{max}^{out}} \right)^{n(\overline{\tau}+1)}$.

Remark. *To adapt the proof to B-strongly connected time-varying directed graphs, one would instead define λ as the maximum Hajnal Coefficient of Ergodicity [35] taken over the product of all possible $(\overline{\tau}+1+B)(\psi+1)$ consecutive matrix products (instead of all $(\overline{\tau}+1)(\psi+1)$ consecutive matrix products). A sufficient assumption in order to prove that $\lambda < 1$ is that a message in transit does not get dropped when the graph topology changes.*

Corollary 1.1 (Convergence to a Neighbourhood for Non-Diminishing Perturbation)**.** *Suppose that the perturbation term is bounded for all $i = 1, 2, \dots, n$; i.e., there exists a positive constant $L < \infty$ such that*

$$\left\| \eta_i[k] \right\|_1 \leq L, \quad \text{for all } i = 1, 2, \dots, n.$$

Then, for all $i = 1, 2, \dots, n$,

$$\lim_{k \to \infty} \left\| z_i^{(0)}[k] - \overline{x}[k] \right\|_1 \leq \frac{CL}{1-q}.$$

Remark 1. *From [37, Lemma 3.1] we know that if $q \in (0,1)$, and $\lim_{s \to \infty} \alpha[s] = 0$, then*

$$\lim_{k \to \infty} \sum_{s=0}^{k} q^{k-s} \alpha[s] = 0.$$

Corollary 1.2 (Exact Convergence for Vanishing Perturbation)**.** *Suppose that the perturbation term vanishes as k (the time-index) tends to infinity, i.e.,*

$$\lim_{k \to \infty} \left\| \boldsymbol{\eta}[\boldsymbol{k}] \right\|_1 = 0,$$

then from the result of Theorem 1 and Remark 1, it holds for all $i = 1, 2, \dots, n$ that

$$\lim_{k \to \infty} \left\| z_i^{(0)}[k] - \overline{x}[k] \right\|_1 = 0.$$

Corollary 1.3 (Geometric Convergence of Asynchronous (Unperturbed) Push-Sum Averaging)**.** *Suppose that for all $i = 1, 2, \dots, n$, and $k \geq 0$, it holds that $\eta_i[k] = \mathbf{0}$. Then from the result of Theorem 1, it holds for all $i = 1, 2, \dots, n$, and $k \geq 0$ that*

$$\left\| z_i^{(0)}[k] - \overline{x}[0] \right\|_1 \leq C q^k \left\| x_i^{(0)}[0] \right\|_1.$$

The proof of Theorem 1 is omitted and can be found in [38]. In brief, the asymptotic product of the asynchronous consensus-matrices, $\overline{\boldsymbol{P}}[k] \cdots \overline{\boldsymbol{P}}[1] \overline{\boldsymbol{P}}[0]$ (for sufficiently large k) is SIA, and furthermore, the entries in the first n rows of the asymptotic product (corresponding to the non-virtual agents) are bounded below by a strictly positive quantity. Applying standard tools from the literature concerning SIA matrices [36] we show that the columns of the asymptotic product of consensus-matrices weakly converge to a stochastic vector sequence at a geometric rate. Substituting this geometric bound into the definition of the asynchronous perturbed Push-Sum updates in Algorithm 2, and after algebraic manipulation similar to that in [21] (which analyzes synchronous delay-free Perturbed Push-Sum), we obtain the desired result.

IV. Asynchronous Gradient-Push

In this section we expound the proposed AGP optimization and present our main convergence results. Our model of asynchrony implies that agents may gossip at different rates, may communicate with arbitrary transmission delays, and may perform gradient steps with stale (outdated) information.

A. Formulation of Asynchronous Gradient-Push

To analyze the AGP optimization algorithm from a global perspective, we use the matrix-based formulation provided in Algorithm 3. At all time-indices k, each agent $v_i^{(r)}$ locally maintains the variables $x_i^{(r)}[k], z_i^{(r)}[k] \in \mathbb{R}^d$, and $y_i^{(r)}[k] \in \mathbb{R}_+$. The *non-virtual* agents initialize these to $z_i^{(0)}[0] = x_i^{(0)}[0] \in \mathbb{R}^d$, and $y_i^{(0)}[0] = 1$. The *virtual* agents' variables are initialized to $z_i^{(r)}[0] = x_i^{(r)}[0] = \mathbf{0}$, and $y_i^{(r)}[0] = 0$ for all $r \neq 0$. This matrix-based formulation describes how the agents' values evolve at those times $t[k+1] \in T = \{t[1], t[2], t[3], \dots, \}$ when one or more agent becomes activated (completes an update). The asynchronous delay-prone communication dynamics are accounted for in the consensus-matrices $\overline{\boldsymbol{P}}[\cdot]$, and the matrix-valued function $\nabla \overline{\boldsymbol{F}}[k+1] \in \mathbb{R}^{n(\overline{\tau}^{\text{msg}}+1) \times d}$ is defined as

$$\nabla \overline{\boldsymbol{F}}[k+1] := \begin{bmatrix} \nabla \boldsymbol{f}^{(0)}(\boldsymbol{z}^{(0)}[k+1]) \\ \mathbf{0} \\ \vdots \\ \mathbf{0} \end{bmatrix},$$

where $\nabla \boldsymbol{f}^{(0)}(\boldsymbol{z}^{(0)}[k+1]) \in \mathbb{R}^{n \times d}$ denotes a block matrix with its i^{th} row equal to

$$\alpha_i[k+1] \delta_i[k+1] \nabla f_i^{(0)}(z_i^{(0)}[k+1]).$$

The scalar $\alpha_i[k+1]$ denotes node v_i's local step-size. The scalar $\delta_i[\cdot]$ is equal to 1 when agent v_i is activated, and equal to 0 otherwise. Recall that agents can only update their local step-sizes when they are activated (i.e., they complete a local gradient step, cf. Algorithm 1). Therefore, if agent v_i is *not* activated at time-index k, then $\alpha_i[k]$ is equal to $\alpha_i[\pi_i(k)]$, the agent's most recently used step-size.[6]

[6]Note: if an agent is not activated at time-index k, then its step-size at that time does have any effect on the execution of the algorithm. We introduce this convention here simply so that the step-size value is well-defined at all times.

Algorithm 3 Asynchronous Gradient Push Optimization

for $k = 0, 1, 2, \ldots$ **to** termination **do**

$$\boldsymbol{x}[k+1] = \overline{\boldsymbol{P}}[k]\left(\boldsymbol{x}[k] - \nabla\overline{\boldsymbol{F}}[k]\right) \tag{9}$$

$$y[k+1] = \overline{\boldsymbol{P}}[k]y[k] \tag{10}$$

$$\boldsymbol{z}[k+1] = \mathrm{diag}(y[k+1])^{-1}\boldsymbol{x}[k+1] \tag{11}$$

B. Main results for Asynchronous Gradient-Push

In this subsection we present the main convergence results for the AGP algorithm.

Assumption 2 (Existence, Convexity, and Smoothness)**.** *Assume that:*

1) *A minimizer of* (1) *exists; i.e.,* $\mathrm{argmin}_x F(x) \neq \emptyset$.
2) *Each function* $f_i(x) : \mathbb{R}^d \to \mathbb{R}$ *is* μ_i*-strongly convex, and has* M_i*-Lipschitz continuous gradients.*

Let $M := \max_i M_i$ and $\mu := \min_i \mu_i$ denote the global Lipschitz constant and modulus of strong convexity, respectively. Let $x^\star := \mathrm{argmin}\, \overline{F}(x)$ denote the global minimizer, and let $x_i^\star := \mathrm{argmin}\, f_i(x)$ denote the minimizer of node v_i's local objective.

Assumption 3 (Step-Size Bound)**.** *Assume that for all agents* v_i*, the terms in the step-size sequence* $\{\alpha_i[k]\}$ *satisfy*

$$\alpha_i[k] \leq \frac{\mu}{2M^2}\left(\frac{1}{N_{max}^{out}}\right)^{n(\overline{\tau}+1)} \qquad \forall k \in \mathbb{N}.$$

Theorem 2 (Bounded Iterates and Gradients)**.** *Suppose Assumptions 2 and 3 are satisfied. Then there exist finite constants* $L, D > 0$ *such that,*

$$\sup_k \|\nabla f_i(z_i[k])\| \leq L, \quad \sup_k \|\overline{x}[k]\| < D.$$

The proof of Theorem 2 appears in [38]. Next we state our main results, the proofs of which all appear in Sec. V. When nodes run asynchronously and at different rates, AGP may not converge precisely to the solution $x^\star$ of (1).

Definition 1 (Re-weighted objective)**.** *Suppose Algorithm 1 is run from time* $t[0]$ *up to time* $t[K]$ *for some integer* $K > 0$*. For all* $i \in [n]$*, let*

$$p_i^{(K)} := \sum_{k=0}^{K-1} \alpha_i[k]\delta_i[k], \quad \textit{and} \quad \overline{p}_i^{(K)} := \frac{p_i^{(K)}}{\sum_{i=1}^n p_i^{(K)}}. \tag{12}$$

Define the re-weighted objective

$$F_K(\cdot) := \sum_{i=1}^n \overline{p}_i^{(K)} f_i(\cdot), \tag{13}$$

and let $x_K^\star$ *denote the minimizer of* $F_K(\cdot)$.

We can characterize how far $x_K^\star$ may be from $x^\star$. Let $\kappa := M/\mu$ denote the condition number of the global objective $F(x)$, let $x_i^\star$ denote the minimizer of $f_i(x)$, let $S_i := \|x_i^\star - x^\star\|$, let $S_{i,j} := \left\|x_i^\star - x_j^\star\right\|$ denote the pairwise distance of agent v_i's minimizer to agent v_j's minimizer, and let $\overline{S} := \max_{i\in[n]} \min_{j\in[n]}(S_{i,j} + S_j)$.

Theorem 3 (Bound on Distance of Minimizers)**.** *Suppose Algorithm 1 is run from time* $t[0]$ *up to time* $t[K]$*, for some integer* $K > 0$*. Let*

$$\Delta^{(K)} := \sqrt{\sum_{i=1}^n \left|\frac{1}{n} - \overline{p}_i^{(K)}\right|}.$$

If Assumption 2 holds, then

$$\|x_K^\star - x^\star\| \leq \frac{\overline{S}\sqrt{\kappa}\,\Delta^{(K)}}{\sqrt{2}},$$

where $\overline{p}_i^{(K)} \in (0,1)$ *and* $x_K^\star$ *are defined in Definition 1, and* $x^\star$ *is the minimizer of* (1)*.*

Theorem 3 bounds the distance between the minimizer of the re-weighted objective (Definition 1) and the minimizer of the original (unbiased) objective (1). The bound depends on the condition number of the global objective, the pairwise distance between agents' local minimizers, the distance between agents' local minimizers and the global (unbiased) minimizer, and the degree of asynchrony in the network. In particular, the quantity $\Delta^{(K)}$ denotes the bias introduced from the processing delays. If agents work at roughly the same rate, then $\Delta^{(K)}$ is close to 0. On the other hand, if there is a large disparity between agents' update rates, then $\Delta^{(K)}$ is close to $\sqrt{2}$.

Assumption 4 (Constant Step-Size)**.** *Suppose Algorithm 1 is run from time* $t[0]$ *up to time* $t[K]$*, for some integer* $K > 0$*. For a given* $\theta \in (0,1)$*, assume that there exist constants* $B > 0$ *and* $w_i \geq 1$*, for all* $i \in [n]$*, such that each agent* v_i *sets its local step-size as*

$$\alpha_i[k] := \alpha_i = \frac{w_i B}{K^\theta}.$$

Note that Assumption 4 prescribes a constant step-size. It reads: first fix the total number of iterations K, and then use K to inform the choice of a constant step-size.[7]

Theorem 4 (Convergence of Asynchronous Gradient Push for Constant Step-Size)**.** *Suppose Algorithm 1 is run from time* $t[0]$ *up to time* $t[K]$*, for some integer* $K > 0$*, and suppose that Assumptions 1, 2, 3, and 4 hold. Then there exist finite positive constants* A_1*,* A_2*, and* A_3 *such that*

$$\frac{1}{K}\sum_{k=0}^{K-1} \|\overline{x}[k] - x_K^\star\|^2 \leq \frac{1}{K^\theta}\left(\frac{n(A_1+A_3)}{2\mu B}\right) + \frac{1}{K}\left(\frac{nA_2}{2\mu B}\right) + \frac{1}{K^{1-\theta}}\left(\frac{n\left(\|\overline{x}[0] - x_K^\star\|^2\right)}{2\mu B}\right),$$

where $\theta \in (0,1)$ *is defined in Assumption 4, and* $x_K^\star$ *is the minimizer of the re-weighted objective defined in Definition 1.*

Explicit expressions for A_1, A_2, and A_3 are given in Lemma 4 below. Both A_2 and A_3 depend on C and q, and hence on the delay bound $\overline{\tau}$.

[7] In practice it may be difficult to determine K ahead of time, since K is the total number of iterations/updates performed *across the entire network*. However in some implementations it may be possible to maintain a (possibly approximate) global count of the number of iterations performed (e.g., by running a separate consensus algorithm in parallel) and use this as a stopping criterion.

Corollary 4.1 (Convergence of Semi-Synchronous Gradient Push for Constant Step-Size)**.** *Suppose the assumptions made in Theorem 4 hold, and suppose that $\overline{\tau}^{proc} = 1$ and each agent v_i sets its local step-size scaling factor $w_i = 1$. Then*

$$\frac{1}{K}\sum_{k=0}^{K-1}\|\overline{x}[k]-x^\star\|^2 \leq \frac{1}{K^\theta}\left(\frac{n(A_1+A_3)}{2\mu B}\right)+\frac{1}{K}\left(\frac{nA_2}{2\mu B}\right) + \frac{1}{K^{1-\theta}}\left(\frac{n\left(\|\overline{x}[0]-x^\star\|^2\right)}{2\mu B}\right),$$

where $x^\star$ is the minimizer of (1)*.*

Corollary 4.1 states that if the agents perform gradient updates at the same rate, then they converge to the unbiased global minimizer, even in the presence of persistent, but bounded, message delays.

Definition 2 (Local iteration counter)**.** *For each agent v_i, and all integers $k \geq 0$, define the local iteration counter*

$$c_i[k] := \sum_{\ell=0}^{k}\delta_i[\ell]$$

to be the number of updates performed by agent v_i in the time-interval $(t[0], t[k]]$. By convention, for all $i \in [n]$, we take $\delta_i[0] := 1$, and thus $c_i[0] = 1$.

Corollary 4.2 (Convergence of Asynchronous Gradient Push for Known Update Rates)**.** *Suppose the assumptions made in Theorem 4 hold, and suppose that each agent v_i has prior knowledge of $c_i[K-1]$, the number of local iterations it will have completed before time $t[K]$. If each agent v_i sets its local step-size scaling factor*

$$w_i := \frac{K}{c_i[K-1]} \geq 1,$$

then

$$\frac{1}{K}\sum_{k=0}^{K-1}\|\overline{x}[k]-x^\star\|^2 \leq + \frac{1}{K^\theta}\left(\frac{n(A_1+A_3)}{2\mu B}\right)+\frac{1}{K}\left(\frac{nA_2}{2\mu B}\right) + \frac{1}{K^{1-\theta}}\left(\frac{n\left(\|\overline{x}[0]-x^\star\|^2\right)}{2\mu B}\right),$$

where $x^\star$ is the minimizer of (1)*.*

Corollary 4.2 states that if the agents know one another's update rates, then they can set their step-sizes to guarantee convergence to the unbiased global minimizer, even in the presence of persistent, but bounded, processing and message delays. In particular, slower agents can simply scale up their step-size to compensate for their slower update rates.

We also provide guarantees for a version of the algorithm using diminishing step sizes.

Assumption 5 (Step-Size Decay)**.** *For a given $\theta \in (0.5, 1)$, assume that there exist constants $B > 0$ and $w_i \geq 1$, for all $i \in [n]$, such that each agent v_i sets its local step-size as*

$$\alpha_i[k] := \frac{w_i B}{(c_i[k])^\theta}.$$

Remark 2. *Note that if Assumption 5 holds, then*

$$\frac{B}{n(k+1)^\theta} \leq \frac{1}{n}\sum_{i=1}^{n}\alpha_i[k]\delta_i[k] \leq \frac{(\frac{1}{n}\sum_{i=1}^{n}w_i)B(\overline{\tau}^{proc})^\theta}{(k+\overline{\tau}^{proc})^\theta},$$

where $\theta \in (0.5, 1)$ is defined in Assumption 5

Theorem 5 (Convergence of Asynchronous Gradient Push for Diminishing Step-Size)**.** *Suppose Algorithm 1 is run from time $t[0]$ up to time $t[K]$, for some integer $K > 0$. If Assumptions 1, 2, 3, and 5 hold, then there exists a finite positive constant A such that*

$$\frac{1}{K}\sum_{k=0}^{K-1}\|\overline{x}[k]-x_K^\star\|^2 \leq \frac{1}{K^{1-\theta}}\left(\frac{n\left(\|\overline{x}[0]-x_K^\star\|^2+A\right)}{2\mu B}\right),$$

where $\theta \in (0.5, 1)$ is defined in Assumption 5, and $x_K^\star$ is the minimizer of the re-weighted objective defined in Definition 1.

Theorem 5 states that in the presence of persistent, but bounded, message and processing delays, the agents converge to the minimizer of a re-weighted version of the original problem, where the re-weighting values are completely determined by the agents' respective cumulative step-sizes during the execution of the algorithm. The constant A depends on the delay bound $\overline{\tau}$; see Lemma 5 below for more details.

Corollary 5.1 (Exact Consensus for Asynchronous Gradient Push)**.** *Suppose the assumptions made in Theorem 5 hold. Then, for all $i \in [n]$,*

$$\lim_{k\to\infty}\|z_i[k]-\overline{x}[k]\| = 0.$$

Proof: Notice that the Asynchronous Gradient Push updates in Algorithm (3) can be regarded as Asynchronous Perturbed Push-Sum updates, with perturbation $\boldsymbol{\eta}[k]$ given by $-\nabla\overline{\boldsymbol{F}}[k]$. Since the gradients remain bounded by Theorem 2, and the local step-sizes go to zero by Assumption 5, the conditions for Corollary 1.3 are satisfied, and it follows that $\lim_{k\to\infty}\|z_i[k]-\overline{x}[k]\| = 0$. ■

Corollary 5.1 states that if all agents use a diminishing step-size, then they will achieve consensus, even in the presence of persistent, but bounded, processing and message delays.

Corollary 5.2 (Convergence of Semi-Synchronous Gradient Push for Diminishing Step-Size)**.** *Suppose the assumptions made in Theorem 5 hold. If $\overline{\tau}^{proc} = 1$ and each agent v_i sets its local step-size scaling factor $w_i = 1$, then*

$$\frac{1}{K}\sum_{k=0}^{K-1}\|\overline{x}[k]-x^\star\|^2 \leq \frac{1}{K^{1-\theta}}\left(\frac{n\left(\|\overline{x}[0]-x^\star\|^2+A\right)}{2\mu B}\right),$$

where $x^\star$ is the minimizer of (1)*.*

Corollary 5.2 states that if the agents perform gradient updates at the same rate, then they converge to the (unbiased) global minimizer, even in the presence of persistent, but bounded, message delays.

Corollary 5.3 (Convergence of Asynchronous Gradient Push for Known Update Rates)**.** *Suppose the assumptions made in Theorem 5 hold, and suppose that each agent v_i has prior*

knowledge of $c_i[K-1]$, the number of local iterations it will have completed before time $t[K]$. If each agent v_i sets its local step-size scaling factor

$$w_i := \frac{\left(\sum_{k=0}^{K-1} \frac{1}{(k+1)^\theta}\right)}{\left(\sum_{k=0}^{c_i[K-1]-1} \frac{1}{(k+1)^\theta}\right)} \geq 1$$

for some $\theta \in (0.5, 1)$ (as per Assumption 5), then

$$\frac{1}{K}\sum_{k=0}^{K-1} \|\overline{x}[k] - x^\star\|^2 \leq \frac{1}{K^{1-\theta}} \left(\frac{n\left(\|\overline{x}[0] - x^\star\|^2 + A\right)}{2\mu B} \right),$$

where $x^\star$ is the minimizer of (1).

Corollary 5.3 states that if the agents know one another's update rates, then they can set their step-sizes to guarantee convergence to the ubiased global minimizer, even in the presence of persistent, but bounded, processing and message delays. In particular, slower agents can simply scale up their step-size to compensate for their slower update rates.

V. Analysis

A. Proof of Theorem 3

Using the strong convexity of the global objective, we have

$$\|x_K^\star - x^\star\|^2 \leq \frac{2}{\mu}\sum_{i=1}^n \frac{1}{n}(f_i(x_K^\star) - f_i(x^\star)), \tag{14}$$

and

$$\|x_K^\star - x^\star\|^2 \leq \frac{2}{\mu}\sum_{i=1}^n \overline{p}_i^{(K)}(f_i(x^\star) - f_i(x_K^\star)). \tag{15}$$

Summing (14) and (15) and multiplying through by $1/2$, we obtain that

$$\|x_K^\star - x^\star\|^2 \leq \frac{1}{\mu}\sum_{i=1}^n \left((f_i(x_K^\star) - f_i(x^\star)) \left(\frac{1}{n} - \overline{p}_i^{(K)} \right) \right).$$

Adding and subtracting $\frac{1}{\mu} f_i(x_i^\star)$, we have

$$\begin{aligned}\|x_K^\star - x^\star\|^2 \leq & \frac{1}{\mu}\sum_{i=1}^n (f_i(x_K^\star) - f_i(x_i^\star)) \left(\frac{1}{n} - \overline{p}_i^{(K)} \right) \\ & - \frac{1}{\mu}\sum_{i=1}^n (f_i(x^\star) - f_i(x_i^\star)) \left(\frac{1}{n} - \overline{p}_i^{(K)} \right).\end{aligned} \tag{16}$$

Define the index set $\mathcal{I} := \{i \in [n] | \frac{1}{n} - \overline{p}_i^{(K)} \geq 0\}$, and its complement $\mathcal{I}^C := \{i \in [n] | \frac{1}{n} - \overline{p}_i^{(K)} < 0\}$. We can further bound (16) as

$$\begin{aligned}\|x_K^\star - x^\star\|^2 \leq & \frac{1}{\mu}\sum_{i\in\mathcal{I}} (f_i(x_K^\star) - f_i(x_i^\star)) \left| \frac{1}{n} - \overline{p}_i^{(K)} \right| \\ & + \frac{1}{\mu}\sum_{i\in\mathcal{I}^C} (f_i(x^\star) - f_i(x_i^\star)) \left| \frac{1}{n} - \overline{p}_i^{(K)} \right|.\end{aligned} \tag{17}$$

Using the smoothness of the global objective, we can bound the terms in the first summation in (17),

$$\frac{1}{\mu}(f_i(x_K^\star) - f_i(x_i^\star)) \left| \frac{1}{n} - \overline{p}_i^{(K)} \right| \leq \frac{\kappa}{2} \|x_K^\star - x_i^\star\|^2 \left| \frac{1}{n} - \overline{p}_i^{(K)} \right|, \tag{18}$$

and similarly for the terms in the second summation in (17),

$$\frac{1}{\mu}(f_i(x^\star) - f_i(x_i^\star)) \left| \frac{1}{n} - \overline{p}_i^{(K)} \right| \leq \frac{\kappa}{2} \|x^\star - x_i^\star\|^2 \left| \frac{1}{n} - \overline{p}_i^{(K)} \right|. \tag{19}$$

Substituting (18) and (19) back into (17), we have

$$\begin{aligned}\|x_K^\star - x^\star\|^2 \leq & \frac{\kappa}{2}\sum_{i\in\mathcal{I}} \|x_K^\star - x_i^\star\|^2 \left| \frac{1}{n} - \overline{p}_i^{(K)} \right| \\ & + \frac{\kappa}{2}\sum_{i\in\mathcal{I}^C} \|x^\star - x_i^\star\|^2 \left| \frac{1}{n} - \overline{p}_i^{(K)} \right|.\end{aligned} \tag{20}$$

Note that there exists an index $j \in [n]$ such that $\left\|x_K^\star - x_j^\star\right\| \leq \left\|x^\star - x_j^\star\right\|$. To see this, suppose for the sake of a contradiction that $\left\|x_K^\star - x_j^\star\right\| > \left\|x^\star - x_j^\star\right\|$ for all $j \in [n]$. Since the local objectives are strongly convex, this implies that there exists a point $x^\star$ such that $f_j(x^\star) < f_j(x_K^\star)$ for all $j \in [n]$. Therefore, $F_K(x^\star) < F_K(x_K^\star)$, which contradicts the definition of $x_K^\star$. Hence there exists $j \in [n]$ such that

$$\left\|x_K^\star - x_j^\star\right\| \leq \left\|x^\star - x_j^\star\right\|. \tag{21}$$

Using the triangle inequality and (21)

$$\|x_K^\star - x_i^\star\| \leq \overline{S}.$$

Similarly, using the triangle inequality

$$\|x_i^\star - x^\star\| \leq \overline{S}.$$

Therefore, we can simplify (20) as

$$\|x_K^\star - x^\star\|^2 \leq \frac{\overline{S}^2\kappa}{2}\sum_{i=1}^n \left| \frac{1}{n} - \overline{p}_i^{(K)} \right|. \tag{22}$$

Taking the square-root on each side of (22) gives the desired result. ■

B. Preliminaries

Before proceeding to the proofs of Theorems 4 and 5, we derive some preliminary results here. Then we give the proof of Theorem 4 followed by the proof of Theorem 5 in the remainder of this section.

Lemma 1. *Suppose Assumptions 2 and 3 are satisfied. Then for all $k \geq 0$,*

$$\|\overline{x}[k] - x_K^\star\| \leq \frac{L}{\mu},$$

where L is defined in Theorem 2, and $x_K^\star$ is the minimizer of the re-weighted objective defined in Definition 1.

Proof: Using the strong convexity of the global objective and the fact that $x_K^\star$ is the minimizer of the re-weighted objective $\sum_{i=1}^n \overline{p}_i^{(K)} f_i(\cdot)$, we have that

$$\|\overline{x}[k] - x_K^\star\| \leq \frac{1}{\mu}\left\| \sum_{i=1}^n \overline{p}_i^{(K)} \nabla f_i(\overline{x}[k]) \right\|.$$

Using the convexity of the norm and substituting the gradient upper bound from Theorem 2 gives the desired result. ∎

Lemma 2. *Suppose Assumptions 2 and 3 are satisfied. Define*

$$\gamma_i[k] := \kappa LC \left\|x_i[0]\right\|_1 q^k$$
$$\chi_i[k] := \kappa L^2 C \sum_{s=0}^{k} q^{k-s} \alpha_i[s]\delta_i[s]$$

where $q \in (0,1)$ and $C > 0$ are defined in Theorem 1. Then for all $i = 1, \ldots, n$ it holds that

$$\begin{aligned}\langle \nabla f_i(z_i[k]), \overline{x}[k] - x_K^\star \rangle \geq\ & \mu \left\|\overline{x}[k] - x_K^\star\right\|^2 \\ & - \gamma_i[k] - \chi_i[k] \\ & + \langle \nabla f_i(x_K^\star), \overline{x}[k] - x_K^\star \rangle.\end{aligned}$$

Proof: Begin by re-writing the inner product

$$\begin{aligned}&\langle \nabla f_i(z_i[k]), \overline{x}[k] - x_K^\star \rangle \\ &\quad = \langle \nabla f_i(z_i[k]) - \nabla f_i(\overline{x}[k]), \overline{x}[k] - x_K^\star \rangle \\ &\qquad + \langle \nabla f_i(\overline{x}[k]), \overline{x}[k] - x_K^\star \rangle.\end{aligned} \tag{23}$$

Using the Lipschitz-smoothness of the objectives, we have

$$\begin{aligned}&\langle \nabla f_i(z_i[k]) - \nabla f_i(\overline{x}[k]), \overline{x}[k] - x_K^\star \rangle \\ &\quad \geq - M \left\|z_i[k] - \overline{x}[k]\right\| \left\|\overline{x}[k] - x_K^\star\right\|.\end{aligned} \tag{24}$$

Making use of Lemma 1, we can simplify (24) as

$$\begin{aligned}&\langle \nabla f_i(z_i[k]) - \nabla f_i(\overline{x}[k]), \overline{x}[k] - x_K^\star \rangle \\ &\quad \geq - \kappa L \left\|z_i[k] - \overline{x}[k]\right\|.\end{aligned} \tag{25}$$

Applying the result of Theorem 1 in (25), and substituting the gradient bounds from Theorem 2, we have

$$\begin{aligned}&\langle \nabla f_i(z_i[k]) - \nabla f_i(\overline{x}[k]), \overline{x}[k] - x_K^\star \rangle \\ &\quad \geq - (\kappa LC) \left(\left\|x_i[0]\right\|_1 q^k + L \sum_{s=0}^{k} q^{k-s} \alpha_i[s]\delta_i[s] \right),\end{aligned}$$

thereby bounding the first term in (23). Using the strong-convexity of the objectives, we can bound the second term in (23) as

$$\begin{aligned}\langle \nabla f_i(\overline{x}[k]), \overline{x}[k] - x_K^\star \rangle \geq\ & \langle \nabla f_i(x_K^\star), \overline{x}[k] - x_K^\star \rangle \\ & + \mu \left\|\overline{x}[k] - x_K^\star\right\|^2.\end{aligned} \tag{26}$$

∎

Lemma 3. *Suppose Assumptions 2 and 3 are satisfied. For any integer $K > 0$, it holds that*

$$\frac{1}{nK} \sum_{k=0}^{K-1} \sum_{i=1}^{n} \alpha_i[k]\delta_i[k] \langle \nabla f_i(x_K^\star), \overline{x}[k] - x_K^\star \rangle \geq 0,$$

where $x_K^\star$ is the minimizer of the re-weighted objective defined in Definition 1.

Proof: Begin by re-writing the inner product

$$\begin{aligned}\langle \nabla f_i(x_K^\star), \overline{x}[k] - x_K^\star \rangle =\ & \langle \nabla f_i(x_K^\star), \overline{x}[K] - x_K^\star \rangle \\ & + \langle \nabla f_i(x_K^\star), \overline{x}[k] - \overline{x}[K] \rangle.\end{aligned} \tag{27}$$

From Lemma 1, we have

$$\begin{aligned}&\frac{1}{nK} \sum_{k=0}^{K-1} \sum_{i=1}^{n} \alpha_i[k]\delta_i[k] \langle \nabla f_i(x_K^\star), \overline{x}[K] - x_K^\star \rangle \\ &\qquad \geq - \frac{L}{\mu} \left\| \frac{1}{nK} \sum_{i=1}^{n} \nabla f_i(x_K^\star) \left(\sum_{k=0}^{K-1} \alpha_i[k]\delta_i[k] \right) \right\|.\end{aligned} \tag{28}$$

Recalling that $p_i^{(K)} := \sum_{k=0}^{K-1} \alpha_i[k]\delta_i[k]$, and that $x_K^\star$ is the minimizer of the re-weighted objective $\sum_{i=1}^{n} f_i(\cdot) p_i^{(K)}$, it follows that the right-hand-side of (28) vanishes, and

$$\frac{1}{nK} \sum_{k=0}^{K-1} \sum_{i=1}^{n} \alpha_i[k]\delta_i[k] \langle \nabla f_i(x_K^\star), \overline{x}[K] - x_K^\star \rangle \geq 0. \tag{29}$$

Now turning our attention to the second term on the right-hand side of (27), we have

$$\begin{aligned}&\langle \nabla f_i(x_K^\star), \overline{x}[k] - \overline{x}[K] \rangle \\ &\qquad = \left\langle \nabla f_i(x_K^\star), \sum_{\ell=k}^{K-1} \frac{1}{n} \sum_{i=1}^{n} \alpha_i[\ell]\delta_i[\ell] \nabla f_i(z_i[\ell]) \right\rangle.\end{aligned}$$

Define the positive integer k' as

$$k' := \operatorname*{argmin}_{k \in \{0,1,\ldots,K-1\}} \left\langle \nabla f_i(x_K^\star), \sum_{\ell=k}^{K-1} \frac{1}{n} \sum_{i=1}^{n} \alpha_i[\ell]\delta_i[\ell] \nabla f_i(z_i[\ell]) \right\rangle,$$

and the corresponding vector, $v_K \in \mathbb{R}^d$,

$$v_K := \sum_{\ell=k'}^{K-1} \frac{1}{n} \sum_{i=1}^{n} \alpha_i[\ell]\delta_i[\ell] \nabla f_i(z_i[\ell]).$$

It holds for all $k = 0, 1, \ldots, K-1$ that

$$\langle \nabla f_i(x_K^\star), \overline{x}[k] - \overline{x}[K] \rangle \geq \langle \nabla f_i(x_K^\star), v_K \rangle.$$

Therefore,

$$\begin{aligned}&\frac{1}{nK} \sum_{k=0}^{K-1} \sum_{i=1}^{n} \alpha_i[k]\delta_i[k] \langle \nabla f_i(x_K^\star), \overline{x}[k] - x^\star[K] \rangle \\ &\qquad \geq - \frac{\|v_K\|}{K} \left\| \frac{1}{n} \sum_{i=1}^{n} \nabla f_i(x_K^\star) \left(\sum_{k=0}^{K-1} \alpha_i[k]\delta_i[k] \right) \right\|.\end{aligned} \tag{30}$$

Note that, from Theorem 2, we have

$$\|v_K\| \leq KL \frac{1}{n} \sum_{i=1}^{n} \alpha_i[0]. \tag{31}$$

Substituting (31) into (30), gives

$$\begin{aligned}&\frac{1}{nK} \sum_{k=0}^{K-1} \sum_{i=1}^{n} \alpha_i[k]\delta_i[k] \langle \nabla f_i(x_K^\star), \overline{x}[k] - x^\star[K] \rangle \\ &\qquad \geq - \left\| \frac{1}{n} \sum_{i=1}^{n} \nabla f_i(x_K^\star) \left(\sum_{k=0}^{K-1} \alpha_i[k]\delta_i[k] \right) \right\| \frac{L}{n} \sum_{i=1}^{n} \alpha_i[0].\end{aligned} \tag{32}$$

Recalling that $p_i^{(K)} := \sum_{k=0}^{K-1} \alpha_i[k]\delta_i[k]$, and that $x_K^\star$ is the minimizer of the re-weighted objective $\sum_{i=1}^n f_i(\cdot)p_i^{(K)}$, it follows that the right-hand side of (32) vanishes, and

$$\frac{1}{nK}\sum_{k=0}^{K-1}\sum_{i=1}^{n}\alpha_i[k]\delta_i[k]\langle\nabla f_i(x_K^\star), \overline{x}[k]-\overline{x}[K]\rangle \geq 0. \quad (33)$$

Summing (33) and (29) together gives the desired result. ∎

Lemma 4. *Suppose Assumptions 2, 3, and 4 are satisfied. Define*

$$b_1[K] := L\sum_{k=0}^{K-1}\left(\frac{1}{n}\sum_{i=1}^{n}\alpha_i[k]\delta_i[k]\right)^2$$
$$b_2[K] := 2L\sum_{k=0}^{K-1}\left(\frac{1}{n}\sum_{i=1}^{n}\alpha_i[k]\delta_i[k]\gamma_i[k]\right)$$
$$b_3[K] := 2L\sum_{k=0}^{K-1}\left(\frac{1}{n}\sum_{i=1}^{n}\alpha_i[k]\delta_i[k]\chi_i[k]\right),$$

where $\gamma_i[k]$ and $\chi_i[k]$ are given in Lemma 2. There exist finite constants $A_1, A_2, A_3 > 0$, such that,

$$b_1[K] \leq \frac{A_1}{K^{2\theta-1}}, \quad b_2[K] \leq \frac{A_2}{K^{\theta}}, \quad b_3[K] \leq \frac{A_3}{K^{2\theta-1}}.$$

Proof: From Assumption 4, we have

$$b_1[K] \leq L\left(\frac{B}{n}\sum_{i=1}^{n}w_i\right)^2\frac{1}{K^{2\theta-1}}.$$

Letting $A_1 := \left(\frac{\sqrt{L}B}{n}\sum_{i=1}^n w_i\right)^2$, we have $b_1[K] \leq \frac{A_1}{K^{2\theta-1}}$. Now to bound $b_2[K]$, note that, given Assumption 4, we have

$$\sum_{k=0}^{K-1}\left(\alpha_i[k]\delta_i[k]\right)q^k \leq \frac{\alpha_i}{1-q}.$$

Letting $A_2 := \frac{2\kappa L^2 C\|x_i[0]\|\left(\frac{B}{n}\sum_{i=1}^n w_i\right)}{(1-q)}$, we have $b_2[K] \leq \frac{A_2}{K^\theta}$. Lastly, to bound $b_3[K]$, it follows from Assumption 4, that

$$\sum_{k=0}^{K-1}\chi_i[k]\left(\alpha_i[k]\delta_i[k]\right) \leq \alpha_i^2\kappa L^2 C\sum_{k=0}^{K-1}\sum_{s=0}^{k}q^{k-s} \leq \frac{\alpha_i^2\kappa L^2 CK}{1-q}.$$

Letting $A_3 := \frac{2\kappa L^3 C\left(\frac{B}{n}\sum_{i=1}^n w_i\right)^2}{(1-q)}$, we have $b_3[K] \leq \frac{A_3}{K^{2\theta-1}}$. ∎

Lemma 5. *Suppose Assumptions 2, 3, and 5 are satisfied. Define*

$$b_1[K] := L\sum_{k=0}^{K-1}\left(\frac{1}{n}\sum_{i=1}^{n}\alpha_i[k]\delta_i[k]\right)^2$$
$$b_2[K] := 2L\sum_{k=0}^{K-1}\left(\frac{1}{n}\sum_{i=1}^{n}\alpha_i[k]\delta_i[k]\gamma_i[k]\right)$$
$$b_3[K] := 2L\sum_{k=0}^{K-1}\left(\frac{1}{n}\sum_{i=1}^{n}\alpha_i[k]\delta_i[k]\chi_i[k]\right),$$

where $\gamma_i[k]$ and $\chi_i[k]$ are given in Lemma 2. There exists a finite constant $A > 0$, such that for all $K \geq 0$,

$$b_1[K] + b_2[K] + b_3[K] \leq A.$$

Proof: First note that the sequences $b_1[K]$, $b_2[K]$, and $b_3[K]$ are all monotonically increasing with K. Therefore, if we can show that the sequences are bounded, then it follows that they are also convergent, and their respective limits serve as upper bounds. From Assumption 5 and Remark 2, it immediately follows that the sequence $b_1[K]$ is bounded, and therefore convergent. Let $A_1' := \lim_{K\to\infty} b_1[K]$. Consequently, $b_1[K] \leq A_1'$ for all $K \geq 0$. Now to bound $b_2[K]$, note that, given Assumption 5, it holds that

$$\sum_{k=0}^{\infty}\left(\alpha_i[k]\delta_i[k]\right)q^k \leq \frac{\alpha_i[0]}{1-q} < \infty.$$

Let $A_2' := \frac{2\kappa L^2 C\|x_i[0]\|}{1-q}\frac{1}{n}\sum_{i=1}^n\alpha_i[0]$. It follows that $b_2[K] \leq A_2'$ for all $K \geq 0$. Lastly, to bound $b_3[K]$, it follows from [37, Lemma 3.1] and Assumption 5, that

$$\sum_{k=0}^{\infty}\chi_i[k]\left(\alpha_i[k]\delta_i[k]\right) \leq \kappa L^2 C\sum_{k=0}^{\infty}\sum_{s=0}^{k}q^{k-s}\left(\alpha_i[s]\delta_i[s]\right)^2 < \infty.$$

Therefore, $b_3[K]$ is bounded and convergent. Let $A_3' := \lim_{K\to\infty} b_3[K]$. Then $b_3[K] \leq A_3' < \infty$ for all $K \geq 0$. Defining $A := A_1' + A_2' + A_3'$ gives the desired result. ∎

C. Proof of Theorem 4

Recall the update equation (9) given by

$$\boldsymbol{x}[k+1] = \overline{\boldsymbol{P}}[k]\left(\boldsymbol{x}[k] - \nabla\overline{\boldsymbol{F}}[k]\right).$$

Since the matrices $\overline{\boldsymbol{P}}[k]$ are column stochastic, we can multiply each side of (9) by $\mathbf{1}^T/n$ to get

$$\overline{x}[k+1] = \overline{x}[k] - \sum_{i=1}^{n}\frac{\alpha_i[k]\delta_i[k]}{n}\nabla f_i(z_i[k]). \quad (34)$$

Subtracting $x_K^\star$ from each side of (35) and taking the squared norm

$$\begin{aligned}\|\overline{x}[k+1] - x_K^\star\|^2 \leq{}& \|\overline{x}[k] - x_K^\star\|^2 \\ &- \frac{2}{n}\sum_{i=1}^{n}\alpha_i[k]\delta_i[k]\langle\nabla f_i(z_i[k]), \overline{x}[k] - x_K^\star\rangle \\ &+ \left\|\frac{1}{n}\sum_{i=1}^{n}\alpha_i[k]\delta_i[k]\nabla f_i(z_i[k])\right\|^2.\end{aligned} \quad (35)$$

Note that, from Theorem 2, we have

$$\left\|\frac{1}{n}\sum_{i=1}^{n}\alpha_i[k]\delta_i[k]\nabla f_i(z_i[k])\right\|^2 \leq \left(\frac{L}{n}\sum_{i=1}^{n}\alpha_i[k]\delta_i[k]\right)^2,$$

thereby bounding the last term in (35). Additionally, making use of Lemma 2, it follows that

$$\begin{aligned}\left\|\overline{x}[k+1]-x_K^\star\right\|^2 \leq \left\|\overline{x}[k]-x_K^\star\right\|^2 &+ \left(\frac{L}{n}\sum_{i=1}^n \alpha_i[k]\delta_i[k]\right)^2 \\ &- 2\mu\left\|\overline{x}[k]-x_K^\star\right\|^2\left(\frac{1}{n}\sum_{i=1}^n \alpha_i[k]\delta_i[k]\right) \\ &- \frac{2}{n}\sum_{i=1}^n \alpha_i[k]\delta_i[k]\langle\nabla f_i(x_K^\star), \overline{x}[k]-x_K^\star\rangle \\ &+ \frac{2}{n}\sum_{i=1}^n \alpha_i[k]\delta_i[k](\gamma_i[k]+\chi_i[k]).\end{aligned} \tag{36}$$

Rearranging terms, averaging each side of (36) across time indices, and making use of Lemma 3 gives

$$\begin{aligned}&\frac{2\mu}{K}\sum_{k=0}^{K-1}\left\|\overline{x}[k]-x_K^\star\right\|^2\left(\frac{1}{n}\sum_{i=1}^n \alpha_i[k]\delta_i[k]\right) \\ &\leq \frac{1}{K}\sum_{k=0}^{K-1}\left(\left\|\overline{x}[k]-x_K^\star\right\|^2 - \left\|\overline{x}[k+1]-x_K^\star\right\|^2\right) \\ &\quad + \frac{1}{K}\sum_{k=0}^{K-1}\left(\frac{2}{n}\sum_{i=1}^n \alpha_i[k]\delta_i[k]\left(\gamma_i[k]+\chi_i[k]\right)\right) \\ &\quad + \frac{1}{K}\sum_{k=0}^{K-1}\left(\frac{L}{n}\sum_{i=1}^n \alpha_i[k]\delta_i[k]\right)^2.\end{aligned} \tag{37}$$

Noticing that we have a telescoping sum on the right hand side of (37), and making use of Lemma 4 and Assumption 4, it follows that

$$\begin{aligned}\frac{1}{K}\sum_{k=0}^{K-1}\left\|\overline{x}[k]-x_K^\star\right\|^2 \leq &\frac{1}{K^{1-\theta}}\left(\frac{n\left(\left\|\overline{x}[0]-x_K^\star\right\|^2\right)}{2\mu B}\right) \\ &+ \frac{1}{K^\theta}\left(\frac{n(A_1+A_3)}{2\mu B}\right) + \frac{1}{K}\left(\frac{nA_2}{2\mu B}\right)\end{aligned}$$

where $\theta \in (0,1)$ is defined in Assumption 4. ■

D. Proof of Corollary 4.1

If $\overline{\tau}^{\text{proc}} = 1$, then each agent performs a gradient update in each iteration. In particular, $\delta_i[k] = 1$ for all $k \geq 0$ and $i = 1,\ldots,n$. Using the fact that $w_i = 1$ for all $i = 1,\ldots,n$ (agents use the same factor in their local step-sizes), it follows that $p_i^{(K)} = p_j^{(K)}$ for all $i,j = 1,\ldots,n$. Hence, the minimizer of the re-weighted objective reduces to that of the original (unbiased) objective, *i.e.*, $x_K^\star = x^\star$. Substituting into the result of Theorem 4 gives the desired result. ■

E. Proof of Corollary 4.2

Note that

$$p_i^{(K)} := \sum_{k=0}^{K-1}\alpha_i[k]\delta_i[k] = \frac{w_i B}{K^\theta}c_i[K-1].$$

Given the choice of w_i, it follows that

$$p_i^{(K)} = \frac{B}{K^{\theta-1}},$$

and is agnostic of the index i. Therefore, $p_i^{(K)} = p_j^{(K)}$ for all $i,j = 1,\ldots,n$. Hence, the minimizer of the re-weighted objective reduces to that of the original (unbiased) objective, *i.e.*, $x_K^\star = x^\star$. Substituting into the result of Theorem 4 gives the desired result. ■

F. Proof of Theorem 5

The proof of Theorem 5 is identical to that of Theorem 4 up to (37). Noticing that we have a telescoping sum on the right hand side of (37), and making use of Lemma 5 and Remark 2, it follows that

$$\frac{1}{K}\sum_{k=0}^{K-1}\left\|\overline{x}[k]-x_K^\star\right\|^2 \leq \quad \frac{1}{K^{1-\theta}}\left(\frac{n\left(\left\|\overline{x}[0]-x_K^\star\right\|^2 + A\right)}{2\mu B}\right),$$

where $\theta \in (0.5,1)$ is defined in Assumption 5. ■

G. Proof of Corollary 5.2

If $\overline{\tau}^{\text{proc}} = 1$, then each agent performs a gradient update in each iteration. In particular, $\delta_i[k] = 1$ for all $k \geq 0$ and $i = 1,\ldots,n$. Using the fact that $w_i = 1$ for all $i = 1,\ldots,n$ (agents use the same factor in their local step-sizes), it follows that $p_i^{(K)} = p_j^{(K)}$ for all $i,j = 1,\ldots,n$. Hence, the minimizer of the re-weighted objective reduces to that of the original (unbiased) objective, *i.e.*, $x_K^\star = x^\star$. Substituting into the result of Theorem 5 gives the desired result. ■

H. Proof of Corollary 5.3

Note that

$$p_i^{(K)} := \sum_{k=0}^{K-1}\alpha_i[k]\delta_i[k] = \sum_{k=0}^{c_i[K-1]-1}\frac{w_i B}{(k+1)^\theta}.$$

Given the choice of w_i, it follows that

$$p_i^{(K)} = \sum_{k=0}^{K-1}\frac{B}{(k+1)^\theta},$$

and is agnostic of the index i. Therefore, $p_i^{(K)} = p_j^{(K)}$ for all $i,j = 1,\ldots,n$. Hence, the minimizer of the re-weighted objective reduces to that of the original (unbiased) objective, *i.e.*, $x_K^\star = x^\star$. Substituting into the result of Theorem 5 gives the desired result. ■

VI. Experiments

Next, we report experiments on a high performance computing cluster. In these experiments, each agent is implemented as a process running on a dedicated CPU core, and each agent runs on a different server. Communication between servers happens over an InfiniBand network. The code to reproduce these experiments is available online;[8] all code is written in Python, and the Open-MPI distribution is used with Python bindings (mpi4py) for message passing.

We report two sets of experiments. The first set involves solving a least-squares regression problem using synthetic

[8] https://github.com/MidoAssran/maopy

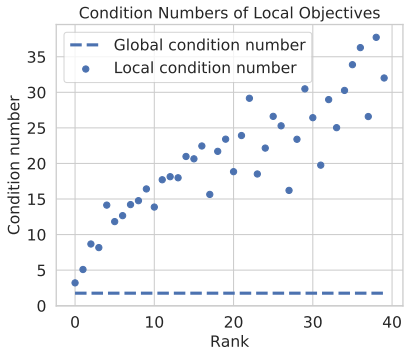


Fig. 3. Condition numbers of local objective functions for a 40-agent partition of the synthetic dataset. The dashed line shows the condition number of the global objective.

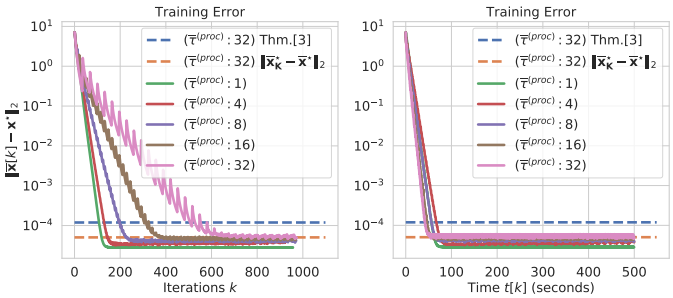


Fig. 4. Convergence of Asynchronous Gradient Push for a 40-agent ring-network with various degrees of asynchrony (quantified by $\overline{\tau}^{\text{proc}}$). The dashed blue bar corresponds to the $\left\|x_K^\star - x^\star\right\|$ bound from Theorem 3, where the reweighing values $\{\overline{p}_i^{(K)}\}$ are computed from the experiment corresponding to $\overline{\tau}^{\text{proc}} = 32$. The dashed orange bar corresponds to the true value of $\left\|x_K^\star - x^\star\right\|$ for the experiments corresponding to $\overline{\tau}^{\text{proc}} = 32$.

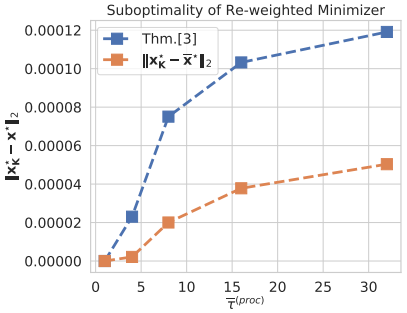


Fig. 5. Distance between the minimizer of the re-weighted objective $x_K^\star$ and the original (unbiased) objective for different choices of $\overline{\tau}^{\text{proc}}$. The blue points depict the bound in Theorem 3, and the red points depict the true quantity.

data. The aim of these experiments is to validate the theory developed in the sections above for AGP. The second set of experiments involves solving a regularized multinomial logistic regression problem on a real dataset. In these experiments we compare AGP with three synchronous methods: Push DIGing (PD) [24], Extra Push (EP) [23], and Synchronous (Sub)Gradient-Push (SGP) [21]. Both PD and EP use gradient tracking to achieve stronger theoretical convergence guarantees at the cost of additional communication overhead. We also compare with Asy-SONATA [39], an asynchronous method that incorporates gradient tracking and which appeared online during the review process of this paper. Note that all methods that use gradient tracking (PD, EP, and Asy-SONATA) require additional memory at each agent and also have a communication overhead per-iteration which is twice that of SGP and AGP.

A. Synthetic Dataset

To validate some of the theory developed in previous sections, we first report experiments on a linear least-squares regression problem using synthetic data. The objective is to minimize, over parameters $\boldsymbol{w}$, the function:

$$F(\boldsymbol{w}) := \frac{1}{D}\sum_{\ell=1}^{D}(w_j^T x^\ell - y^\ell)^2, \tag{38}$$

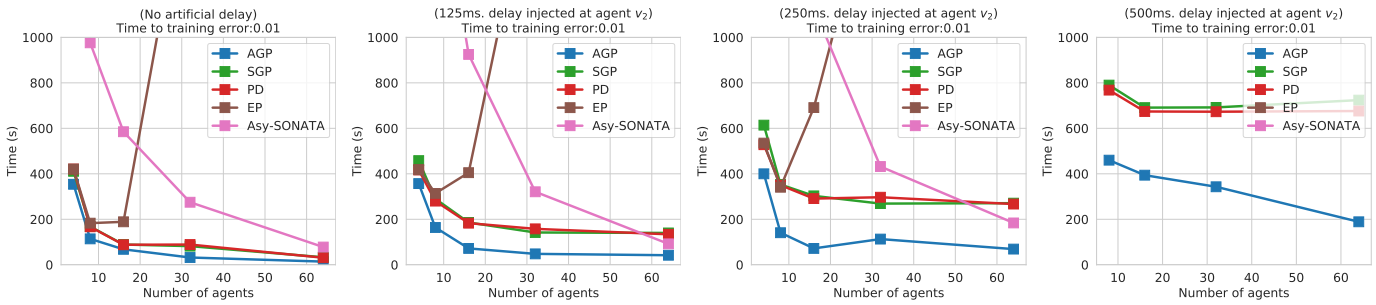
where $D = 2{,}560{,}000$ is the number of training instances in the dataset, $x^l \in \mathbb{R}^{50}$ and $y^l \in \mathbb{R}^1$ correspond to the l^{th} training instance feature and label vectors respectively, and $\boldsymbol{w} \in \mathbb{R}^{50}$ are the model parameters. We generate the data $\{(x^\ell, y^\ell)\}_{\ell=1}^D$ using the technique suggested in [40].

The D data samples are partitioned among the n agents. The local objective function f_i at agent v_i is similar to that in (38) but the sum over l only involves those training instances assigned to agent v_i. The condition number of the global objective is approximately 2. The condition number of individual agents' local objectives is diverse and depends on the data-partition. Figure 3 shows the local objective conditioning for a 40-agent partition of the dataset. The condition numbers of the local objectives are approximately uniformly spaced in the interval $(3, 37)$.

During training, agent v_i logs the values of z_i and the time after every update. Post training, we analytically compute the minimizer of the re-weighted objective defined in Definition 1. To validate the bound on the distance between the minimizer of the re-weighted objective and the original unbiased objective (cf. Theorem 3), we run AGP for different choices of $\overline{\tau}^{\text{proc}}$. We control $\overline{\tau}^{\text{proc}}$ by forcing an agent to block if it completes $\overline{\tau}^{\text{proc}}$ iterations while another agent still hasn't completed a single iteration in the same time interval; thus, in the worst case scenario, a fast agent can complete $\overline{\tau}^{\text{proc}}$ iterations for every iteration completed by a slow agent.[9] In Fig. 4 we show the convergence of AGP for different values of $\overline{\tau}^{\text{proc}}$. We use a directed ring network in this example to examine the worst-case scenario.

Increasing $\overline{\tau}^{\text{proc}}$ leads to a reduction in the iteration-wise convergence rate, as expected. However, increasing $\overline{\tau}^{\text{proc}}$ also reduces the idling time, and thereby leads to an improvement in the time-wise convergence rate. The dashed blue line in Fig. 4 corresponds to the upper bound on $\|x_K^\star - x^\star\|$ from

[9]For the purpose of this experiment, we artificially delay half of the agents in the network by 500 ms each iteration, and implement $\overline{\tau}^{\text{proc}}$ programmatically using non-blocking barrier operations (which are a part of the MPI-3 standard). In particular, each agent tests a non-blocking barrier request at each local iteration. If the test is passed, then a new non-blocking barrier request object is created. If the test is not passed and more than $\overline{\tau}^{\text{proc}}$ local iterations have gone by since the last test was passed, then the agent blocks and waits for the barrier-test to pass. In this way, no more than $\overline{\tau}^{\text{proc}}$ iterations can be performed by the network in the time it takes any single agent to complete one local iteration.



(a) No artificial delay (b) 125ms delay injected at agent v_2 (c) 250ms delay injected at agent v_2 (d) 500ms delay injected at agent v_2

Fig. 6. Time $t[k]$ (seconds) at which $F(\overline{x}[k]) - F(x^\star) < 0.01$ is satisfied for the first time in the Covertype experiments. (a) Experiment run under normal operating conditions. (b) An artificial 125ms delay is injected at agent v_2 after every local iteration. (b) An artificial 250ms delay is injected at agent v_2 after every local iteration. (c) An artificial 500ms delay is injected at agent v_2 after every local iteration; neither EP nor Asy-SONATA obtained a residual error of 10^{-2} or below after 1000s for this delay with any network size. AGP reaches the threshold residual error 10^{-2} faster than all other methods.

Theorem 3, where the values $\overline{p}_i^{(K)}$ are computed from the experiment corresponding to $\overline{\tau}^{\text{proc}} = 32$. The dashed orange line corresponds to the true value of $\|x_K^\star - x^\star\|$, where the values $\overline{p}_i^{(K)}$ are also computed from the experiment corresponding to $\overline{\tau}^{\text{proc}} = 32$.

In Fig. 5 we plot the distance between the minimizer of the re-weighted objective and the original (unbiased) objective for each of the different choices of $\overline{\tau}^{\text{proc}}$ used in this experiment. As predicted from Theorem 3, the distance between minimizers decreases as the disparity in agent update rates decreases.

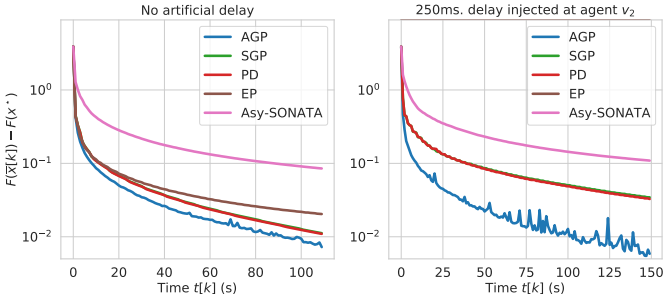
B. Non-Synthetic Dataset

To facilitate comparisons with existing methods in the literature, a regularized multinomial logistic regression classifier is trained on the *Covertype* dataset [41] from the UCI repository [42]. Here the objective is to minimize, over model parameters $\boldsymbol{w}$, the negative log-likelihood loss function:

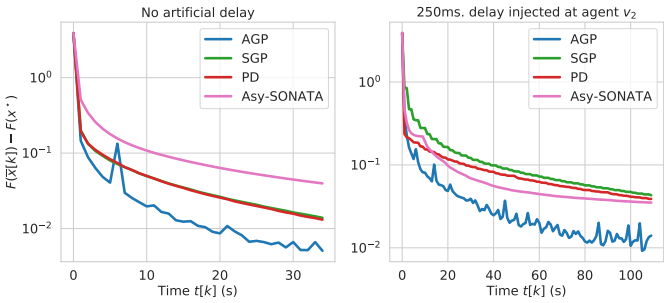
$$F(\boldsymbol{w}) := -\sum_{l=1}^{D}\sum_{j=1}^{K} \log\left(\frac{\exp(w_j^T x^l)}{\sum_{j'=1}^{K}\exp(w_{j'}^T x^l)}\right)^{y_j^l} + \frac{\lambda}{2}\|\boldsymbol{w}\|_{\boldsymbol{F}}^2, \tag{39}$$

where $D = 581{,}012$ is the number of training instances in the dataset, $K = 7$ is the number of classes, $x^l \in \mathbb{R}^{54}$ and $y^l \in \mathbb{R}^7$ correspond to the l^{th} training instance feature and label vectors respectively (the label vectors are represented using a 1-hot encoding), $\boldsymbol{w} \in \mathbb{R}^{7\times 54}$ are the model parameters, and $\lambda > 0$ is a regularization parameter. We take $\lambda = 10^{-4}$ in the experiments. The 54 features consist of a mix of categorical (binary 1 or 0) features and real numbers. We whiten the non-categorical features by subtracting the mean and dividing by the standard deviation.

All network topologies are randomly generated using the Erdős-Rényi model where the expected out-degree of each agent is 4, independent of n; *i.e.*, with an edge probability of $\min\{4/(n-1), 1\}$. To investigate how the algorithms scale with the number of nodes, we consider different values of $n \in \{4, 8, 16, 32, 64\}$. In each case, we randomly partition the D training instances evenly across the n agents. All algorithms use a constant step-size, and we tuned the step-sizes separately for each algorithm using a simple grid-search over the range $\alpha \in [10^{-3}, 10^{1}]$. For all algorithms, the (constant) step-size



(a) 16-agent Erdős-Rényi graph



(b) 64-agent Erdős-Rényi graph

Fig. 7. Multinomial logistic regression training error on the Covertype dataset using large multi-agent networks. Left subplots in each figure correspond to normal operating conditions. Right subplots correspond to experiments with an artificial 250ms delay induced at agent v_2 at each local iteration. (EP did not converge over the 64-agent network topology). AGP is more robust than the synchronous algorithms to failing or stalling nodes.

$\alpha = 1.0$ gave the best performance. Since the total number of samples D is fixed, this problem has a fixed computational workload; as we increase the size of the network, the number of samples (and hence, the computational load) per agent decreases. The local objective function f_i at agent v_i is similar to that in (39) but the sum over l only involves those training instances assigned to agent v_i.

Fig. 6 shows the first time $t[k]$ when the residual error satisfies $F(\overline{x}[k]) - F(x^\star) < 0.01$, as a function of network size. Fig. 6a shows that, under normal operating conditions, AGP decreases the residual error for both small and large network sizes faster than the state-of-the art methods and its synchronous counterpart. To study robustness of the methods

TABLE I
AVERAGE TIME TAKEN BY AN AGENT TO PERFORM A GRADIENT-BASED UPDATE FOR THE COVERTYPE EXPERIMENTS.

# agents	Mean time (s)	Max.. time (s)	Min.. time (s)
4	0.362 ± 0.00649	0.507	0.348
8	0.0993 ± 0.0107	0.139	0.0859
16	0.0488 ± 0.00339	0.0598	0.0430
32	0.0207 ± 0.00166	0.0284	0.0175
64	0.00849 ± 0.000246	0.0123	0.00797

to delays, we run experiments where we inject an artificial delay at agent v_2 after every local iteration; the results are shown in Fig. 6b, Fig. 6c, and Fig. 6d for 125 ms, 250 ms, and 500ms delays, respectively. To put the magnitude of these delays in context, Table I reports the average agent update time for various network sizes. As expected, we observe that asynchronous algorithms (AGP and Asy-SONATA) are more robust than the synchronous algorithms to slow nodes. However, for the 500 ms delay case, Asy-SONATA did not achieve a residual error below 0.01 after 1000 seconds. Fig. 6d demonstrates that AGP is robust to such a large delay.

Fig. 7 shows the residual error curves with respect to wall clock time for different network sizes, with and without an artificial 250ms delay induced at agent v_2 at each iteration. AGP is faster than the other methods under normal operating conditions (left subplots Fig. 7), and this performance improvement is especially pronounced when an artificial 250ms delay is injected in the network (right subplots Fig. 7). In the smaller multi-agent networks, a 250ms delay is a relatively plausible occurrence. In larger multi-agent networks a 250ms delay is quite extreme since there could be over 2000 updates performed by the network in the time it takes the artificially delayed agent to compute a single update. The fact that AGP is still able to converge in this scenario is a testament to its robustness.

VII. CONCLUSION

Our analysis of asynchronous Gradient-Push handles communication and computation delays. We believe our results could be extended to also deal with dropped messages using the approach described in [43], in which dropped messages appear as additional communication delays, which are easily addressed in our analysis framework.

Corollary 5.3 showed that when agents know their relative update rates, then asynchronous Gradient-Push can be made to converge to the minimizer of f rather than that of the reweighted objective (13) by appropriately scaling the step-size. After the initial preprint of this work appeared online [44], a related method was proposed in [45] to estimate and track the update rates in a decentralized manner at the cost of additional communication overhead. Another related method was proposed in [39] that uses gradient tracking in combination with two sets of robust, asynchronous averaging updates — one row stochastic, the other column stochastic — to achieve provably geometric convergence rates at the cost of additional communication overhead and storage at each agent.

While extending synchronous Gradient-Push to an asynchronous implementation has produced considerable performance improvements, it remains the case that Gradient-Push is simply a multi-agent analog of gradient descent, and it would be interesting to explore the possibility of extending other algorithms to asynchronous operation using singly-stochastic consensus matrices; *e.g.*, exploring methods that use an extrapolation between iterates to accelerate convergence; or quasi-Newton methods that approximate the Hessian using only first-order information; or Lagrangian-dual methods that formulate the consensus constrained optimization problems using the Lagrangian, or Augmented Lagrangian, and simultaneously solve for both primal and dual variables. Furthermore, it would be interesting to establish convergence rates for asynchronous versions of these algorithms.

Lastly, we find that, in practice, agents can asynchronously and independently control the upper bound on their relative processing delays, $\overline{\tau}^{\text{proc}}$, by using non-blocking barrier primitives, such as those available as part of the MPI-3 standard. It may be interesting to explore treating this as an algorithm parameter, rather than something dictated by the environment, and decreasing the delay bound according to some local iteration schedule so that one can realize the speed advantages of asynchronous methods at the start of training, and obtain the benefits of synchronous methods as one approaches the minimizer. For example, from Definition 1, it is clear that $\|x_K^\star - x^\star\| \to 0$ when $\overline{\tau}^{\text{proc}} \to 0$. We believe that this is another interesting direction of future work.

REFERENCES

[1] A. Nedić, A. Olshevsky, and M. G. Rabbat, "Network topology and communication-computation tradeoffs in decentralized optimization," *Proceedings of the IEEE*, vol. 106, no. 5, pp. 953–976, 2018.

[2] D. P. Bertsekas and J. N. Tsitsiklis, *Parallel and distributed computation: numerical methods*. Prentice hall Englewood Cliffs, NJ, 1989, vol. 23.

[3] K. Tsianos, S. Lawlor, and M. G. Rabbat, "Communication/computation tradeoffs in consensus-based distributed optimization," in *Advances in neural information processing systems*, 2012, pp. 1943–1951.

[4] K. I. Tsianos, S. Lawlor, and M. G. Rabbat, "Consensus-based distributed optimization: Practical issues and applications in large-scale machine learning," in *Proceedings of the 50th Annual Allerton Conference on Communication, Control, and Computing*. IEEE, 2012, pp. 1543–1550.

[5] D. Kempe, A. Dobra, and J. Gehrke, "Gossip-based computation of aggregate information," in *Proceedings of the 44th Annual IEEE Symposium on Foundations of Computer Science.* IEEE, 2003, pp. 482–491.

[6] J. Dean and L. A. Barroso, "The tail at scale," *Communications of the ACM*, vol. 56, no. 2, pp. 74–80, 2013.

[7] M. G. Rabbat and K. I. Tsianos, "Asynchronous decentralized optimization in heterogeneous systems," in *Proceedings of the 53rd IEEE Annual Conference on Decision and Control*. IEEE, 2014, pp. 1125–1130.

[8] X. Lian, W. Zhang, C. Zhang, and J. Liu, "Asynchronous decentralized parallel stochastic gradient descent," in *International Conference on Machine Learning*, 2018, pp. 3049–3058.

[9] M. Assran and M. Rabbat, "An empirical comparison of multi-agent optimization algorithms," in *Proceedings of the IEEE Global Conference on Signal and Information Processing*. IEEE, 2017, pp. 573–577.

[10] L. Cannelli, F. Facchinei, V. Kungurtsev, and G. Scutari, "Asynchronous parallel algorithms for nonconvex big-data optimization. part ii: Complexity and numerical results," *arXiv preprint arXiv:1701.04900*, 2017.

[11] M. T. Hale, A. Nedić, and M. Egerstedt, "Asynchronous multi-agent primal-dual optimization," *IEEE Transactions on Automatic Control*, vol. 62, no. 9, pp. 4421–4435, 2017.

[12] S. Kumar, R. Jain, and K. Rajawat, "Asynchronous optimization over heterogeneous networks via consensus admm," *IEEE Transactions on Signal and Information Processing over Networks*, vol. 3, no. 1, pp. 114–129, 2017.

[13] A. Aytekin, "Asynchronous algorithms for large-scale optimization: Analysis and implementation," Ph.D. dissertation, KTH Royal Institute of Technology, 2017.

[14] T. Wu, K. Yuan, Q. Ling, W. Yin, and A. H. Sayed, "Decentralized consensus optimization with asynchrony and delays," in *Proceedings of the 50th Asilomar Conference on Signals, Systems and Computers*. IEEE, 2016, pp. 992–996.

[15] W. Gropp, E. Lusk, N. Doss, and A. Skjellum, "A high-performance, portable implementation of the mpi message passing interface standard," *Parallel computing*, vol. 22, no. 6, pp. 789–828, 1996.

[16] A. Nedić and A. Ozdaglar, "On the rate of convergence of distributed subgradient methods for multi-agent optimization," in *Proceedings of the 46th IEEE Conference on Decision and Control*. IEEE, 2007, pp. 4711–4716.

[17] F. Bénézit, V. Blondel, P. Thiran, J. Tsitsiklis, and M. Vetterli, "Weighted gossip: Distributed averaging using non-doubly stochastic matrices," in *Proceedings of the IEEE International Symposium on Information Theory*. IEEE, 2010, pp. 1753–1757.

[18] T. Charalambous, Y. Yuan, T. Yang, W. Pan, C. N. Hadjicostis, and M. Johansson, "Distributed finite-time average consensus in digraphs in the presence of time delays," *IEEE Transactions on Control of Network Systems*, vol. 2, no. 4, pp. 370–381, 2015.

[19] C. N. Hadjicostis and T. Charalambous, "Average consensus in the presence of delays in directed graph topologies," *IEEE Transactions on Automatic Control*, vol. 59, no. 3, pp. 763–768, 2014.

[20] K. I. Tsianos, S. Lawlor, and M. G. Rabbat, "Push-sum distributed dual averaging for convex optimization," in *Proceedings of the 51st IEEE Conference on Decision and Control*, 2012, pp. 5453–5458.

[21] A. Nedić and A. Olshevsky, "Distributed optimization over time-varying directed graphs," *IEEE Transactions on Automatic Control*, vol. 60, no. 3, pp. 601–615, 2015.

[22] C. Xi and U. A. Khan, "Dextra: A fast algorithm for optimization over directed graphs," *IEEE Transactions on Automatic Control*, vol. 62, no. 10, pp. 4980–4993, 2017.

[23] J. Zeng and W. Yin, "Extrapush for convex smooth decentralized optimization over directed networks," *Journal of Computational Mathematics*, vol. 35, no. 4, pp. 383–396, 2017.

[24] A. Nedić, A. Olshevsky, and W. Shi, "Achieving geometric convergence for distributed optimization over time-varying graphs," *SIAM Journal on Optimization*, vol. 27, no. 4, pp. 2597–2633, 2017.

[25] J. Tsitsiklis, D. Bertsekas, and M. Athans, "Distributed asynchronous deterministic and stochastic gradient optimization algorithms," *IEEE Transactions on Automatic Control*, vol. 31, no. 9, pp. 803–812, 1986.

[26] S. Li and T. Basar, "Asymptotic agreement and convergence of asynchronous stochastic algorithms," *IEEE Transactions on Automatic Control*, vol. 32, no. 7, pp. 612–618, 1987.

[27] M. Eisen, A. Mokhtari, and A. Ribeiro, "Decentralized quasi-newton methods," *IEEE Transactions on Signal Processing*, vol. 65, no. 10, pp. 2613–2628, 2017.

[28] F. Mansoori and E. Wei, "Superlinearly convergent asynchronous distributed network newton method," *IEEE 56th Annual Conference on Decision and Control (CDC)*, pp. 2874–2879, 2017.

[29] S. Boyd, A. Ghosh, B. Prabhakar, and D. Shah, "Randomized gossip algorithms," *IEEE/ACM Transactions on Networking (TON)*, vol. 14, no. SI, pp. 2508–2530, 2006.

[30] A. G. Dimakis, S. Kar, J. M. Moura, M. G. Rabbat, and A. Scaglione, "Gossip algorithms for distributed signal processing," *Proceedings of the IEEE*, vol. 98, no. 11, pp. 1847–1864, 2010.

[31] A. Nedić, "Asynchronous broadcast-based convex optimization over a network," *IEEE Transactions on Automatic Control*, vol. 56, no. 6, pp. 1337–1351, 2011.

[32] F. Iutzeler, P. Bianchi, P. Ciblat, and W. Hachem, "Asynchronous distributed optimization using a randomized alternating direction method of multipliers," in *Proceedings of the 52nd IEEE Annual Conference on Decision and Control*. IEEE, 2013, pp. 3671–3676.

[33] E. Wei and A. Ozdaglar, "On the o (1= k) convergence of asynchronous distributed alternating direction method of multipliers," in *Proceedings of the IEEE Global Conference on Signal and Information Processing*. IEEE, 2013, pp. 551–554.

[34] P. Bianchi, W. Hachem, and F. Iutzeler, "A coordinate descent primal-dual algorithm and application to distributed asynchronous optimization," *IEEE Transactions on Automatic Control*, vol. 61, no. 10, pp. 2947–2957, 2016.

[35] J. Hajnal and M. Bartlett, "Weak ergodicity in non-homogeneous markov chains," *Mathematical Proceedings of the Cambridge Philosophical Society*, vol. 54, no. 2, pp. 233–246, 1958.

[36] J. Wolfowitz, "Products of indecomposable, aperiodic, stochastic matrices," *Proceedings of the American Mathematical Society*, vol. 14, no. 5, pp. 733–737, 1963.

[37] S. S. Ram, A. Nedić, and V. V. Veeravalli, "Distributed stochastic subgradient projection algorithms for convex optimization," *Journal of optimization theory and applications*, vol. 147, no. 3, pp. 516–545, 2010.

[38] M. Assran, "Asynchronous subgradient push: Fast, robust, and scalable multi-agent optimization," Master's thesis, McGill University, 2018.

[39] Y. Tian, Y. Sun, and G. Scutari, "Achieving linear convergence in distributed asynchronous multi-agent optimization," *arXiv preprint arxiv:1803.10359*, March 2018.

[40] M. L. Lenard and M. Minkoff, "Randomly generated test problems for positive definite quadratic programming," *ACM Transactions on Mathematical Software (TOMS)*, vol. 10, no. 1, pp. 86–96, 1984.

[41] J. A. Blackard and D. J. Dean, "Comparative accuracies of artificial neural networks and discriminant analysis in predicting forest cover types from cartographic variables," *Computers and Electronics in Agriculture*, vol. 24, no. 3, pp. 131–151, 2000.

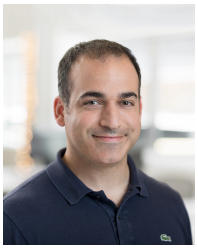
[42] D. Dua and C. Graff, "UCI machine learning repository," Irvine, CA, 2019. [Online]. Available: http://archive.ics.uci.edu/ml

[43] C. N. Hadjicostis, N. H. Vaidya, and A. D. Dominguez-Garcia, "Robust distributed average consensus via exchange of running sums," *IEEE Transactions on Automatic Control*, vol. 61, no. 6, pp. 1492–1507, Jun. 2016.

[44] M. Assran and M. G. Rabbat, "Asynchronous subgradient-push," March 2018, arXiv preprint arXiv:1803.08950v1.

[45] J. Zhang and K. You, "AsySPA: An exact asynchronous algorithm for convex optimization over digraphs," Aug. 2018, arxiv preprint https://arxiv.org/abs/1808.04118.

Mahmoud S. Assran, ("Mido") received the B.Eng. degree and M.Eng. degree in honours electrical engineering from McGill University in 2017 and 2018 respectively. He is currently pursuing a Ph.D. at McGill University under the supervision of Prof. Michael Rabbat, and is also a research assistant at Facebook Artificial Intelligence Research. Mido is a Vadasz Doctoral Fellow in Engineering and is the recipient of a Graduate Excellence Fellowship, the Accenture Prize in Engineering and Science, the (Intel) Les Vadasz Award in Engineering, an NSERC-USRA award, and is a 2017 Rhodes Scholar Finalist. His research interests include multi-agent optimization and applications thereof in machine learning contexts. In particular, Mido is interested in using multi-agent approaches to develop computationally efficient learning algorithms.



Michael G. Rabbat (S'02–M'07–SM'15) received the B.Sc. degree from the University of Illinois, Urbana-Champaign, in 2001, the M.Sc. degree from Rice University, Houston, TX, in 2003, and the Ph.D. degree from the University of Wisconsin, Madison, in 2006, all in electrical engineering. He is a Research Scientist with Facebook Artificial Intelligence Research. From 2007–2018 he was a professor in the Department of Electrical and Computer Engineering at McGill University. During the 2013–2014 academic year he held visiting positions at Télécom Bretegne, Brest, France, the Inria Bretagne-Atlantique Reserch Centre, Rennes, France, and KTH Royal Institute of Technology, Stockholm, Sweden. He previously served on the editorial boards of IEEE TRANSACTIONS ON CONTROL OF NETWORK SYSTEMS, IEEE SIGNAL PROCESSING LETTERS, and IEEE TRANSACTIONS ON SIGNAL AND INFORMATION PROCESSING OVER NETWORKS. His research interests include distributed algorithms for optimization and inference, graph signal processing, and applications in large-scale machine learning and statistical signal processing.