Distributed Gradient Methods with Variable Number of Working Nodes

Dušan Jakovetić, Dragana Bajović, Nataša Krejić, and Nataša Krklec-Jerinkić

***Abstract*—We consider distributed optimization where N nodes in a connected network minimize the sum of their local costs subject to a common constraint set. We propose a distributed projected gradient method where each node, at each iteration k, performs an update (is active) with probability p_k, and stays idle (is inactive) with probability $1-p_k$. Whenever active, each node performs an update by weight-averaging its solution estimate with the estimates of its active neighbors, taking a negative gradient step with respect to its local cost, and performing a projection onto the constraint set; inactive nodes perform no updates. Assuming that nodes' local costs are strongly convex, with Lipschitz continuous gradients, we show that, as long as activation probability p_k grows to one asymptotically, our algorithm converges in the mean square sense (MSS) to the same solution as the standard distributed gradient method, i.e., as if all the nodes were active at all iterations. Moreover, when p_k grows to one linearly, with an appropriately set convergence factor, the algorithm has a linear MSS convergence, with practically the same factor as the standard distributed gradient method. Simulations demonstrate that, when compared with the standard distributed gradient method, the proposed algorithm significantly reduces the overall number of per-node communications and per-node gradient evaluations (computational cost) for the same required accuracy.**

***Index Terms*—Distributed optimization, distributed gradient method, variable number of working nodes, convergence rate, consensus.**

I. Introduction

We consider distributed optimization where N nodes constitute a generic, connected network, each node i has a convex cost function $f_i : \mathbb{R}^d \mapsto \mathbb{R}$ known only by i, and the nodes want to solve the following problem:

$$\begin{array}{ll} \text{minimize} & \sum_{i=1}^N f_i(x) =: f(x) \\ \text{subject to} & x \in \mathcal{X}, \end{array} \tag{1}$$

where $x \in \mathbb{R}^d$ is the optimization variable common to all nodes, and $\mathcal{X} \subset \mathbb{R}^d$ is a closed, convex constraint set, known by all. The above and related problems arise frequently, e.g., in big data analytics in cluster or cloud environments, e.g., [1]-[3], distributed estimation in wireless sensor networks (WSNs), e.g., [4]-[8], and distributed control applications, e.g., [9], [10]. With all the above applications, data is split across multiple networked nodes (sensors, cluster machines, etc.), and $f_i(x) = f_i(x; D_i)$ represents a loss with respect to data D_i stored locally at node i.

A popular approach to solve (1) is via distributed (projected) (sub)gradient methods, e.g., [11], [12], [13]. With these methods, each node i, at each iteration k, updates its solution estimate by weight-averaging it with the estimates of its neighbors, taking a negative gradient step with respect to its local cost, and projecting the result onto the constraint set $\mathcal{X}$. Distributed gradient methods are attractive as they do not require centralized coordination, have inexpensive iterations (provided that projections onto $\mathcal{X}$ are computationally light), and exhibit resilience to inter-node communication failures and delays; however, they have a drawback of slow convergence rate.

Several techniques to improve convergence rates of distributed (projected) gradient methods have been proposed, including Newton-like methods, e.g., [14], [15], [16], [17], and Nesterov-like methods, e.g., [18], [19]. In this paper, we make distributed (projected) gradient methods more efficient by proposing a novel method with a variable number of working nodes. Each node i, at each iteration k, performs an update (is active) with probability p_k, and stays idle (is inactive) with probability $1-p_k$. Whenever active, each node i performs the same update as with the standard distributed gradient method, while inactive nodes perform no updates.

Our main results are as follows. Assuming that the costs f_i's are strongly convex and their gradients are Lipschitz continuous, we show that, whenever the activation probability p_k grows asymptotically to one, our method converges in the mean square sense to the same point as the standard distributed gradient method.[1] Moreover, when p_k grows to one linearly, with the convergence factor $\delta \in (0,1)$, our algorithm has a linear convergence rate (in the sense of the expected distance to the solution). When, in addition, quantity δ is set in accordance with the f_i's condition number and the underlying network's spectral gap, we show that the proposed algorithm converges practically with the same linear convergence factor as the standard distributed gradient method (albeit with a larger hidden constant). Hence, interestingly, our algorithm achieves practically the same rate in iterations k as the standard distributed gradient method, but with the reduced cost per iteration k (overall communication and computational cost), thus making distributed gradient methods more efficient. Simulation examples on l_2-regularized logistic losses confirm

D. Jakovetić and D. Bajović are with University of Novi Sad, BioSense Institute, Novi Sad, Serbia. N. Krejić and N. Krklec-Jerinkić are with Department of Mathematics and Informatics, Faculty of Science, University of Novi Sad, Novi Sad, Serbia. Research of N. Krejić and N. Krklec-Jerinkić is supported by Ministry of Education, Science and Technological Development, Republic of Serbia, grant no. 174030. Authors' e-mails: [djakovet,dbajovic]@uns.ac.rs, natasak@uns.ac.rs, natasa.krklec@dmi.uns.ac.rs.

[1]Under a constant step-size α, the standard (projected) distributed gradient method converges to a point in a neighborhood of the solution of (1), where the corresponding squared distance is $O(\alpha)$; see ahead Theorem 1 and, e.g., [20], [21].

that our method significantly reduces the communication and computational cost with respect to the standard distributed gradient method, for the same desired accuracy.

The communication and computational savings are highly relevant with applications like WSNs and distributed learning in cluster or cloud environments. With WSNs, the reduced communication and computational cost to retrieve the result translate into energy saving of the sensor motes' batteries and the increase of the network lifetime. With distributed learning in cluster or cloud environments, less amount of communication and computation for a specific application/task means that the saved resources can be re-allocated to another concurrent tasks. For example, at times when a node with our method is idle, the resources allocated to it (e.g., a virtual cloud machine) can be released and re-allocated to other tasks.

We explain intuitively the above results that we achieve. Namely, standard distributed gradient method exhibits, in a sense, two sources of redundancy–the first corresponds to the inter-node communications aspect, while the second corresponds to an optimization aspect (number of gradient evaluations per iteration) of the algorithm. It turns out that, as we show here, a careful simultaneous exploitation of these two redundancies allows to match the rate of the standard distributed gradient method with a reduced "work." The two sources of redundancy have been already noted in the literature, but have not been exploited simultaneously before. The communication redundancy, e.g., [22] means that the inter-node communications can be "sparsified," e.g., through the intermittent link failures, so that the algorithm still remains convergent. In other words, it is not necessary to utilize communications through all the available links at all iterations for the algorithm to converge. The optimization redundancy has been previously studied only in the context of centralized optimization, e.g., [23]. The core idea is that, under certain assumptions on the cost functions, a (centralized) stochastic-type gradient method with an appropriately increasing sample size matches the convergence rate of the standard gradient method with the full sample size at all iterations, as shown in [23].

A. Related work

We now briefly review existing work relevant to our contributions to help us further contrast our work from the literature. We divide the literature into two classes: 1) distributed gradient methods for multi-agent optimization; and 2) centralized stochastic approximation methods with variable sample sizes. The former class relates to our work through the communication redundancy, while the latter considers the optimization redundancy.

Distributed gradient methods for multi-agent optimization. Distributed methods of this type date back at least to the 80s, e.g., [24], and have received renewed interest in the past decade, e.g., [11]. Reference [11] proposes the distributed (sub)gradient method with a constant step-size, and analyzes its performance under time-varying communication networks. Reference [22] considers distributed (sub)gradient method under random communication networks with failing links and establishes almost sure convergence under a diminishing step-size rule. A major difference of our paper from the above works is that, in [24], [11], [22], only inter-node communications over iterations are "sparsified," while each node performs gradient evaluations at each iteration k. In [13], the authors propose a gossip-like scheme where, at each k, only two neighboring nodes in the network wake up and perform weight-averaging (communication among them) and the negative gradient step with respect to their respective local costs, while the remaining nodes stay idle. The key difference with respect to our paper is that, with our method, the number of active nodes over iterations k (on average) is increasing, while in [13] it remains equal to two for all k. Consequently, the established convergence properties of the two methods are very different.

There have been many works where nodes or links in the network are controlled by random variables. References [25], [26], [27], [28] consider distributed algorithms for solving the consensus problem – finding an average of nodes' local scalars a_i's, while we consider here a more general problem (1). These consensus algorithms involve only local averaging steps, and no local gradient steps (while we have here both local averaging and local gradient steps). The models of averaging, i.e., weight matrices, which [25], [26], [27], [28] assume are very different from ours: they all assume random weight matrices with time-invariant distributions, while ours are time-varying. Reference [29] proposes a control mechanism for link activations in diffusion algorithms to minimize the estimation error under given resource constraints. The main differences with respect to our paper are that [29] assumes that local gradients are always incorporated (deterministic step-sizes), and the link activation probabilities are time invariant.

References [30], [31], [32] provide a thorough and in-depth analysis of diffusion algorithms under a very general model of asynchrony, where both the combination (weight) matrices and nodes' step-sizes are random. Our work differs from these references in several aspects, which include the following. A major difference is that papers [30], [31], [32] assume that both the step sizes'and the combination matrices' random processes have constant (time-invariant) first and second moments, and the two processes are moreover mutually independent. In contrast, both our weight matrices and step-sizes are random, with time-varying distributions. Actually, the fact that, with our method node activation probabilities converge to one (which corresponds to time-varying first moment of the step-sizes) is critical to establish our main results (Theorems 2 and 3). As a consequence of the different assumed settings, the results here and in [30], [31], [32] are also qualitatively different. Namely, [30], [31], [32] show that asynchronous diffusion has a similar convergence rate as synchronous diffusion, while the steady state error (expected squared distance of a node's estimate from the solution) is degraded (although moderately so). Namely, the steady state error of asynchronous diffusion is $O(\mu)$ (μ is the step-size) larger than with the synchronous diffusion. Hence, their difference is of the same order as the steady state error itself – $O(\mu)$; see, e.g., [32], page 25. On the other hand, we show here that, when node activation probabilities increase along iterations sufficiently fast to unity, both the convergence rate and the steady state error are not degraded. Further differences are that papers [30], [31], [32] allow for noisy gradients, their nodes' local cost functions all have the same minimizers, and therein the optimization problem is unconstrained. In contrast, we assume noise-free gradients, different local minimizers, and constrained problems.

Our paper is also related to reference [33], which considers diffusion algorithms with two types of nodes – informed and uninformed. The informed nodes both: 1) acquire measurements and perform in-network processing (which translates into computing gradients in our scenario); and 2) perform consultation with neighbors (which translates into weight-averaging the estimates across neighborhoods), while the uninformed nodes only perform the latter task. The authors study the effect of the proportion of informed nodes and their distribution in space. A key difference with respect to our work is that the uninformed nodes in [33] still perform weight-averaging, while the idle nodes here perform no processing. Finally, we comment on reference [34] which introduces an adaptive policy for each node to decide whether it will communicate with its neighbors or not and demonstrate significant savings in communications with respect to the always-communicating scenario. A major difference of [34] from our paper is that, with [34], nodes always perform local gradients, i.e., they do not stay idle (in the sense defined here).

Centralized stochastic approximation methods with variable sample sizes have been studied for a long time. We distinguish two types of methods: the ones that assume unbounded sample sizes (where the cost function is in the form of a mathematical expectation) and the methods with bounded sample sizes (where the cost function is of the form in (1).) Our work contrasts with both of these threads of works by considering distributed optimization over an arbitrary connected network, while they consider centralized methods.

Unbounded sample sizes have been studied, e.g., in [35], [36], [37], [38], [39]. Reference [35] uses a Bayesian scheme to determine the sample size at each iteration within the trust region framework, and it shows almost sure convergence to a problem solution. Reference [36] shows almost sure convergence as long as the sample size grows sufficiently fast along iterations. In [37], the variable sample size strategy is obtained as the solution of an associated auxiliary optimization problem. Further references on careful analyses of the increasing sample sizes are, e.g., [38], [39].

References [40], [41] consider a trust region framework and assume bounded sample sizes, but, differently from our paper and [23], [35], [37], [38], [39], they allow the sample size both to increase and to decrease at each iteration. The paper chooses a sample size at each iteration such that a balance is achieved between the decrease of the cost function and the width of an associated confidence interval. Reference [42] proposes a schedule sequence in the monotone line search framework which also allows the sample size both increase and decrease at each iteration; paper [43] extends the results in [42] to a non-monotone line search.

Reference [23] is closest to our paper within this thread of works, and our work mainly draws inspiration from it. The authors consider a bounded sample size, as we do here. They consider both deterministic and stochastic sampling and determine the increase of the sample size along iterations such that the algorithm attains (almost) the same rate as if the full sample size was used at all iterations. A major difference of [23] with respect to the current paper is that they are not concerned with the networked scenario, i.e., therein a central entity works with the variable (increasing) sample size. This setup is very different from ours as it has no problem dimension of propagating information across the networked nodes – the dimension present in distributed multi-agent optimization.

Paper organization. The next paragraph introduces notation. Section II explains the model that we assume and presents our proposed distributed algorithm. Section III states our main results which we prove in Section IV. Section V provides numerical examples. Finally, we conclude in Section VI.

Notation. We denote by: $\mathbb{R}$ the set of real numbers; $\mathbb{R}^d$ the d-dimensional Euclidean real coordinate space; A_{ij} the entry in the i-th row and j-th column of a matrix A; $A^\top$ the transpose of a matrix A; $\odot$ and $\otimes$ the Hadamard (entry-wise) and Kronecker product of matrices, respectively; I, 0, $\mathbf{1}$, and e_i, respectively, the identity matrix, the zero matrix, the column vector with unit entries, and the i-th column of I; J the $N \times N$ matrix $J := (1/N)\mathbf{1}\mathbf{1}^\top$; $A \succ 0\,(A \succeq 0)$ means that the symmetric matrix A is positive definite (respectively, positive semi-definite); $\|\cdot\|_l$ the vector (respectively, matrix) l-norm of its vector (respectively, matrix) argument; $\|\cdot\| = \|\cdot\|_2$ the Euclidean (respectively, spectral) norm of its vector (respectively, matrix) argument; $\lambda_i(\cdot)$ the i-th largest eigenvalue, $\mathrm{Diag}\,(a)$ the diagonal matrix with the diagonal equal to the vector a; $|\cdot|$ the cardinality of a set; $\nabla h(w)$ the gradient evaluated at w of a function $h : \mathbb{R}^d \to \mathbb{R}$, $d \geq 1$; $\mathbb{P}(\mathcal{A})$ and $\mathbb{E}[u]$ the probability of an event $\mathcal{A}$ and expectation of a random variable u, respectively. For two positive sequences η_n and χ_n, we have: $\eta_n = O(\chi_n)$ if $\limsup_{n\to\infty} \frac{\eta_n}{\chi_n} < \infty$.

II. Model and algorithm

Subsection II-A describes the optimization and network models that we assume, while Subsection II-B presents our proposed distributed algorithm with variable number of working nodes.

A. Problem model

Optimization model. We consider optimization problem (1), and we impose the following assumptions on (1).

Assumption 1 (Optimization model) (a) For all i, $f_i : \mathbb{R}^d \mapsto \mathbb{R}$ is strongly convex with modulus $\mu > 0$, i.e.:

$$f_i(y) \geq f_i(x) + \nabla f_i(x)^\top (y-x) + \frac{\mu}{2}\|y-x\|^2, \forall x, y \in \mathbb{R}^d.$$

(b) For all i, $f_i : \mathbb{R}^d \mapsto \mathbb{R}$ has Lipschitz continuous gradient with constant L, $0 < \mu \leq L < \infty$, i.e.:

$$\|\nabla f_i(x) - \nabla f_i(y)\| \leq L\|x-y\|, \quad \forall x, y \in \mathbb{R}^d.$$

(c) The set $\mathcal{X} \subset \mathbb{R}^d$ is nonempty, closed, convex, and bounded.

We denote by $D := \max\{\|x\| : x \in \mathcal{X}\}$ the diameter of $\mathcal{X}$. Note that, as $\mathcal{X}$ is compact, the gradients $\nabla f_i(x)$'s are bounded over $x \in \mathcal{X}$, i.e., there exists $G > 0$, such that, for all i, for all $x \in \mathcal{X}$, $\|\nabla f_i(x)\| \le G$. The constant G can be taken as $LD + \max_{i=1,...,N} \|\nabla f_i(0)\|$. Indeed, for any $x \in \mathcal{X}$, we have:

$$\begin{aligned} \|\nabla f_i(x)\| &\le \|\nabla f_i(x) - \nabla f_i(0)\| + \|\nabla f_i(0)\| \\ &\le L\|x\| + \|\nabla f_i(0)\| \\ &\le LD + \max_{i=1,...,N} \|\nabla f_i(0)\|. \end{aligned}$$

Similarly, there exist constants $-\infty < m_f \le M_f < \infty$, such that $m_f \le f_i(x) \le M_f$, $\forall i$, $\forall x \in \mathcal{X}$. Constants m_f and M_f can be taken as $M_f = -m_f = G\,D + \max_{i=1,...,N} |f_i(0)|$. Under Assumption 1, (1) is solvable and has a unique solution, which we denote by $x^\star$.

Network model. Nodes are connected in a generic undirected network $\mathcal{G} = (\mathcal{V}, E)$, where $\mathcal{V}$ is the set of N nodes and E is the set of edges – all node (unordered) pairs $\{i,j\}$ that can exchange messages through a communication link. We impose the following assumption.

Assumption 2 (Network connectedness) The network $\mathcal{G} = (\mathcal{V}, E)$ is connected, undirected, and simple (no self-loops nor multiple links).

Both Assumptions 1 and 2 hold throughout the paper. We denote by Ω_i the neighborhood set of node i (excluding i). We associate with $\mathcal{G}$ a $N \times N$ symmetric weight matrix C, which is also stochastic (rows sum to one and all the entries are non-negative). We let C_{ij} be strictly positive for each $\{i,j\} \in E$, $i \ne j$; $C_{ij} = 0$ for $\{i,j\} \notin E$, $i \ne j$; and $C_{ii} = 1 - \sum_{j \ne i} C_{ij}$. As we will see, the weights C_{ij}'s will play a role in our distributed algorithm. The quantities C_{ij}, $j \in \Omega_i$, are assumed available to node i before execution of the distributed algorithm. We assume that matrix C has strictly positive diagonal entries (each node assigns a non-zero weight to itself) and is positive definite, i.e., $\lambda_N(C) > 0$. For a given arbitrary stochastic, symmetric weight matrix C' with positive diagonal elements, positive definiteness may not hold. However, such arbitrary C' can be easily adapted to generate matrix C that obeys all the required properties (symmetric, stochastic, positive diagonal elements), and, in addition, is positive definite. Namely, letting, for some $\kappa \in (0,1)$, $C := \frac{\kappa+1}{2} I + \frac{1-\kappa}{2} C'$, we obtain that $\lambda_N(C) > \kappa$. It can be shown that, under the above assumptions on C, $\lambda_1(C) = 1$, and $\lambda_2(C) < 1$.

B. Proposed distributed algorithm

We now describe the distributed algorithm to solve (1) that we propose. We assume that all nodes are synchronized according to a global clock and simultaneously (in parallel) perform iterations $k = 0, 1, ...$ At each iteration k, each node i updates its solution estimate $x_i^{(k)} \in \mathcal{X}$, with arbitrary initialization $x_i^{(0)} \in \mathcal{X}$. To avoid notational clutter, we will assume that $x_i^{(0)} = x_j^{(0)}$, $\forall i, j$. Further, each node has an internal Bernoulli state variable $z_i^{(k)}$. If $z_i^{(k)} = 1$, node i updates $x_i(k)$ at iteration k; we say that, in this case, node i is active at k. If $z_i^{(k)} = 0$, node i keeps its current state $x_i(k)$ and does not perform an update; we say that, in this case, node i is idle. At each k, each node i generates $z_i^{(k)}$ independently from the previous iterations, and independently from other nodes. We denote by $p_k := \mathbb{P}\left(z_i(k) = 1\right)$. The quantity p_k is our algorithm's tuning parameter, and is common for all nodes. We assume that, for all k, $p_k \ge p_{\min}$, for a positive constant $p_{\min}$.

Denote by $\Omega_i^{(k)}$ the set of working neighbors of node i at k, i.e., all nodes $j \in \Omega_i$ with $z_j^{(k)} = 1$. The update of node i is as follows. If $z_i^{(k)} = 0$, node i is idle and sets $x_i^{(k+1)} = x_i^{(k)}$. Otherwise, if $z_i^{(k)} = 1$, node i broadcasts its state to all its working neighbors $j \in \Omega_i^{(k)}$. The non-working (idle) neighbors do not receive $x_i^{(k)}$; for example, with WSNs, this corresponds to switching-off the receiving antenna of a node. Likewise, node i receives $x_j^{(k)}$ from all $j \in \Omega_i^{(k)}$. Upon reception, node i updates $x_i^{(k)}$ as follows:

$$\begin{aligned} x_i^{(k+1)} &= \mathcal{P}_{\mathcal{X}} \left\{ \left(1 - \sum_{j \in \Omega_i^{(k)}} C_{ij} \right) x_i^{(k)} \right. \qquad (2) \\ &\quad \left. + \sum_{j \in \Omega_i^{(k)}} C_{ij}\, x_j^{(k)} - \frac{\alpha}{p_k} \nabla f_i(x_i^{(k)}) \right\}. \end{aligned}$$

In (2), $\mathcal{P}_{\mathcal{X}}(y) = \arg\min_{v \in \mathcal{X}} \|v - y\|$ denotes the Euclidean projection of point y on $\mathcal{X}$, and $\alpha > 0$ is a constant; we let $\alpha \le \lambda_N(C)/L$. (See ahead Remark 2.) In words, (2) means that node i makes a convex combination of its own estimate with the estimates of its working neighbors, takes a step in the negative direction of its local gradient, and projects the resulting value onto the constraint set. As we will see, multiplying the step-size in (2) by $1/p_k$ compensates for non-working (idle) nodes over iterations.

Remark 1 Setting $p_k = 1$, $\forall k$, corresponds to the standard distributed (sub)gradient method in [44].

Compact representation. We present (2) in a compact form. Denote by $x^{(k)} := \left((x_1^{(k)})^\top, ..., (x_N^{(k)})^\top \right)^\top$, and $z^{(k)} := (z_1^{(k)}, ..., z_N^{(k)})^\top$. Further, introduce $F : \mathbb{R}^{N\,d} \mapsto \mathbb{R}$, with

$$F(x) = F(x_1, ..., x_N) := \sum_{i=1}^N f_i(x_i).$$

Also, denote by $\mathcal{X}^N \subset \mathbb{R}^{N\,d}$ the Cartesian product $\mathcal{X} \times ... \times \mathcal{X}$, where $\mathcal{X}$ is repeated N times. Next, introduce the $N \times N$ random matrix $W^{(k)}$, defined as follows:

$$W_{ij}^{(k)} = \begin{cases} C_{ij} z_i^{(k)} z_j^{(k)} & \text{for } \{i,j\} \in E,\ i \ne j \\ 0 & \text{for } \{i,j\} \notin E,\ i \ne j \\ 1 - \sum_{s \ne i} W_{is}^{(k)} & \text{for } i = j. \end{cases}$$

Then, it is easy to see that, for $k = 0, 1, ...$, update rule (2) can be written as:

$$\begin{aligned} x^{(k+1)} &= \mathcal{P}_{\mathcal{X}^N} \Big\{ (W^{(k)} \otimes I\,)\, x^{(k)} \qquad (3) \\ &- \frac{\alpha}{p_k} \Big(\nabla F(x^{(k)}) \odot (z^{(k)} \otimes \mathbf{1}) \Big) \Big\}, \end{aligned}$$

where $W^{(k)} \otimes I$ denotes the Kronecker product of $W^{(k)}$ and the $d \times d$ identity matrix, $\mathbf{1}$ in (3) is of size $d \times 1$, and $\odot$ denotes the Hadamard (entry-wise) product. Note that sequence $\{x^{(k)}\}$ is a sequence of random vectors, due to the randomness of the $z^{(k)}$'s. The case $p_k \equiv 1$, $\forall k$, corresponds to standard distributed (sub)gradient method in [11], in which case (3) becomes:

$$x^{(k+1)} = \mathcal{P}_{\mathcal{X}^N} \Big\{ \,(\, C \otimes I\,)\; x^{(k)} - \alpha \, \nabla F(x^{(k)}) \Big\}. \qquad (4)$$

III. Statement of main results

We now present our main results on the proposed distributed method (2). For benchmarking of (2), we first present a result on the convergence of standard distributed gradient algorithm (4). All the results in the current section, together with needed auxiliary results, are proved in Section IV. Recall that $x^\star \in \mathbb{R}^d$ is the solution to (1).

Theorem 1 Consider standard distributed gradient algorithm (4) with step-size $\alpha \leq \lambda_N(C)/L$. Then, $x^{(k)}$ converges to a point $x^\bullet = (\,(x_1^\bullet)^\top, ..., (x_N^\bullet)^\top\,)^\top \in \mathcal{X}^N$ that satisfies, for all $i = 1, ..., N$:

$$\begin{aligned} \|x_i^\bullet - x^\star\|^2 &\leq \|x^\bullet - \mathbf{1} \otimes x^\star\|^2 \leq \alpha\, \mathcal{C}_\Psi \qquad (5) \\ \mathcal{C}_\Psi &:= \frac{4N(M_f - m_f)}{1 - \lambda_2(C)} + \frac{2N^2 G^2}{\mu\,(1 - \lambda_2(C))}. \qquad (6) \end{aligned}$$

Furthermore:

$$\|x^{(k)} - x^\bullet\| \leq 2\,\sqrt{N}\, D\, (1 - \alpha\, \mu)^k = O\left((1 - \alpha\, \mu)^k \right). \qquad (7)$$

Theorem 1 says that, with algorithm (4), each node's estimate $x_i^{(k)}$ converges to a point $x_i^\bullet$ in the neighborhood of the true solution $x^\star$; the distance of the limit $x_i^\bullet$ from $x^\star$ is controlled by step-size α – the smaller the step-size, the closer the limit to the true solution. Furthermore, $x_i^{(k)}$ converges to a solution neighborhood (to $x_i^\bullet$) at a globally linear rate, equal to $1 - \alpha\mu$. Hence, there is a tradeoff with respect to the choice of α: a small α means a higher precision in the limit, but a slower rate to reach this precision. Note also that, for $\alpha \leq \lambda_N(C)/L$, the convergence factor $(1 - \alpha\, \mu)$ does not depend on the underlying network, but the distance $\|x_i^\bullet - x^\star\|$ between arbitrary node i's limit $x_i^\bullet$ and the solution $x^\star$ depends on the underlying network – through the number of nodes N and the second largest eigenvalue of matrix C.

Remark 2 It is possible to extend Theorem 1 to allow also for the step-sizes $\alpha \in (\lambda_N(C)/L,\, (1 + \lambda_N(C))/L]$, in which case the convergence factor $(1 - \alpha\, \mu)$ in (5) is replaced with $\max\{\alpha L - 1, 1 - \alpha\, \mu\}$. We restrict ourselves to the case $\alpha \leq \lambda_N(C)/L$, both for simplicity and due to the fact that step-sizes α – needed to achieve sufficient accuracies in practice – are usually much smaller than $1/L$. (See also Section V.)

Remark 3 For $\alpha \leq \lambda_N(C)/L$, the convergence factor $(1 - \alpha\, \mu)$ is an exact (tight) worst case convergence factor, in the following sense: given an arbitrary network and matrix C, and given an arbitrary step-size $\alpha \leq \lambda_N(C)/L$, there exists a specific choice of functions f_i's, set $\mathcal{X}$, and initial point $x^{(0)} \in \mathcal{X}^N$, such that $\|x^{(k+1)} - x^\bullet\| = (1 - \alpha\, \mu)\|x^{(k)} - x^\bullet\|$, for all $k = 0, 1, ...$[2]

We benchmark the proposed method against the standard distributed gradient method by checking: 1) whether it converges to the same point $x^\bullet$; 2) if so, whether it converges linearly; and 3) if the convergence is linear, how the corresponding convergence factor compares with $(1 - \alpha\, \mu)$ – the convergence factor of the standard distributed gradient method.

References [20], [21] also analyze the convergence rate of the standard distributed gradient method, allowing for step-size ranges wider than $\alpha \in (0,\, \lambda_N(C)/L]$. They establish bounds on quantity $\|x^{(k)} - \mathbf{1} \otimes x^\star\|$ which are in general different than (7), and they are not directly concerned with quantity $\|x^{(k)} - x^\bullet\|$, i.e., precise characterization of convergence rate of $x^{(k)}$ towards its limit. We adopt here (7) as it gives an exact worst-case characterization of the convergence rate towards $x^\bullet$ for $\alpha \in (0,\, \lambda_N(C)/L]$ (see Remark 3).

We now state our main results on the proposed algorithm (2). The first result deals with a more generic sequence of the p_k's that converge to one; the second result is for the p_k's that converge to one geometrically.

Theorem 2 Consider algorithm (2) with step-size $\alpha \leq \lambda_N(C)/L$. Further, suppose that $p_k \geq p_{\min}$, $\forall k$, for some $p_{\min} > 0$, and let $p_k \to 1$ as $k \to \infty$. Then, with algorithm (2), the iterates $x^{(k)}$ converge, in the mean square sense, to the same point $x^\bullet$ as the standard distributed gradient method (4), i.e., $\mathbb{E}\left[\|x^{(k)} - x^\bullet\|^2\right] \to 0$ as $k \to \infty$. Assume, in addition, that $p_k = 1 - u_k$, with:

$$0 \leq u_k \leq \frac{\mathcal{C}_u}{(k+1)^{1+\zeta}},\ \forall k,$$

for some constants $\mathcal{C}_u > 0$ and $\zeta > 0$. Then, $x^{(k)}$ converges to $x^\bullet$ almost surely.

Theorem 3 Consider algorithm (2) with step-size $\alpha \leq \lambda_N(C)/L$. Further, suppose that $p_k = 1 - \delta^{k+1}$, $k = 0, 1, ...$, for some $\delta \in (0, 1)$, and let $\eta := \max\{1 - \alpha\mu, \delta^{1/2}\}$. Then, in the mean square sense, algorithm (2) converges to the same point $x^\bullet$ as the standard distributed gradient method (4), and, moreover:

$$\mathbb{E}\left[\|x^{(k)} - x^\bullet\|\right] = O\left(k \eta^k\right) = O\left((\eta + \epsilon)^k\right),$$

for arbitrarily small positive ϵ. Furthermore, if $\sqrt{\delta} \leq 1 - \alpha\mu$:

$$\mathbb{E}\left[\|x^{(k)} - x^\bullet\|\right] = O\left(k (1 - \alpha\mu)^k\right) = O\left((1 - \alpha\mu + \epsilon)^k\right).$$

Theorem 2 states that, provided that the p_k's are uniformly bounded away from zero, from below, and $p_k \to 1$, the

[2]Consider $f_i : \mathbb{R} \to \mathbb{R}$, $f_i(x) = x^2$, $\forall i$, $\mathcal{X} = \{x \in \mathbb{R} : |x| \leq 2\}$, and $x^{(0)} = 1$. Note that, in this case, $x^\bullet = 0$, and $\mu = L = 1$. For this example, it is easy to show that $\|x^{(k+1)} - x^\bullet\| = (1 - \alpha)\|x^{(k)} - x^\bullet\|$, for all $k = 0, 1, ...$, and so the convergence factor equals $1 - \alpha\, \mu$.

method (4) converges (in the mean square) to the same point as the standard distributed method (4). If, moreover, p_k converges to one at a sublinear rate at least $\frac{1}{(k+1)^{1+\zeta}}$ (where $\zeta > 0$ can be arbitrarily small), then the convergence also hold almost surely. Therefore, in such scenarios, the random idling schedule governed by the p_k's does not affect the method's limit.

Theorem 3 furthermore suggests that, provided that convergence o p_k towards unity is linear (geometric) with convergence factor $\delta \leq (1-\alpha\mu)^2$, algorithm (2) converges at the practically same rate as the standard method (4), i.e., as if all the nodes were working all the time (albeit with a larger hidden constant). Hence, we may expect that the proposed method (2) achieves the same desired accuracy as (4) with less amount of resources spent (smaller number of the overall node activations–communications and computations). This indeed occurs in practice, as confirmed by simulations in Section V. The hidden convergence constant is dependent on the underlying network, the sequence $\{p_k\}$, and step-size α, and is given explicitly in Remark 5.

IV. Intermediate results and proofs

Subsection IV-A gives intermediate results on the random matrices $W^{(k)}$ and provides the disagreement estimates – how far apart are the estimates $x_i^{(k)}$ of different nodes in the network. Subsection IV-B introduces a penalty-like interpretation of algorithm (4) and proves Theorem 1. Finally, Subsection IV-C proves our main results, Theorems 2 and 3, by applying the penalty-like interpretation on algorithm (3). For notational simplicity, this section presents auxiliary results and all the proofs for the case $d=1$, but all these extend to a generic $d>1$. Throughout this Section, all the claims (equalities and inequalities) which deal with random quantities hold either: 1) surely, for any random realization; or 2) in expectation. It is clear from notation which of the two cases is in force.

A. Matrices $W^{(k)}$ and disagreement estimates

Matrices $W^{(k)}$. Recall that $J := (1/N)\mathbf{1}\mathbf{1}^\top$. We have the following Lemma on the matrices $W^{(k)}$. Lemma 4 follows from simple arguments and standard results on symmetric, stochastic matrices (see, e.g., [45]). Hence, we omit the proof for brevity.

Lemma 4 (Matrices $W^{(k)}$) (a) The sequence $\{W^{(k)}\}$ is a sequence of independent random matrices.

(b) For all k, $W^{(k)}$ is symmetric and stochastic (rows sum to one and all the entries are nonnegative).

(c) For all k, $0 \prec W^{(k)} \preceq I$.

(d) There exists a constant $\beta \in (0,1)$ such that, $\forall k$, $\mathbb{E}\left[\|W^{(k)} - J\|^2\right] < \beta^2$.

It can be shown that β can be taken as $\beta^2 = 1 - (p_{\min})^N\left[1-(\lambda_2(C))^2\right]$; see, e.g., [45].

Remark 4 The quantities $\mathbb{E}\left[\|W^{(k)}-J\|^2\right]$ clearly depend on k, and, more specifically, on p_k. We adopt here a (possibly loose) uniform bound β (independent of k) as this suffices to establish conclusions about convergence rates of algorithm (3) while simplifying the presentation.

Disagreement estimate. Denote by $\overline{x}^{(k)} := \frac{1}{N}\sum_{i=1}^N x_i^{(k)}$ the global average of the nodes' estimates, and by $\widetilde{x}_i^{(k)} = x_i^{(k)} - \overline{x}^{(k)}$. Note that both quantities are random. The quantity $\widetilde{x}_i^{(k)}$ measures how far is the node i's estimate from the global average. Denote by $\widetilde{x}^{(k)} := (\widetilde{x}_1^{(k)}, ..., \widetilde{x}_N^{(k)})^\top$. The next Lemma shows that $\mathbb{E}\left[\|\widetilde{x}^{(k)}\|^2\right]$ is uniformly bounded, $\forall k$, and that the bound is $O(\alpha^2)$, i.e., the disagreement size is controlled by the step-size. (The smaller the step-size, the smaller the disagreements are.)

Lemma 5 (Disagreements bound) For all k, there holds:

$$\mathbb{E}\left[\|\widetilde{x}^{(k)}\|^2\right] \leq \left(\frac{3\alpha\sqrt{N}G}{p_{\min}(1-\beta)}\right)^2.$$

B. Analysis of the standard distributed gradient method through a penalty-like reformulation

We analyze the proposed method (3) through a penalty-like interpretation, to our best knowledge first introduced in [46]. Introduce an auxiliary function $\Psi_\alpha : \mathbb{R}^N \mapsto \mathbb{R}$, defined by: $\Psi_\alpha(x) := \sum_{i=1}^N f_i(x_i) + \frac{1}{2\alpha}x^\top(I-C)x = F(x) + \frac{1}{2\alpha}x^\top(I-C)x$, and the associated optimization problem:

$$\begin{array}{ll}\text{minimize} & \Psi_\alpha(x) = \sum_{i=1}^N f_i(x_i) + \frac{1}{2\alpha}x^\top(I-C)x \\ \text{subject to} & x \in \mathcal{X}^N.\end{array} \quad (8)$$

Function Ψ_α and (8) will be very useful in the analysis of (2). In fact, we will show that (2) is an inexact version of the (projected) gradient method on function Ψ_α. Clearly, (8) is solvable, and it has a unique solution, which we denote by $x^\bullet$.[3]

We start by showing that standard distributed (sub)gradient method in [11] is an exact (projected) gradient method on Ψ_α. Indeed, the derivative $\nabla\Psi_\alpha(x) = \nabla F(x) + \frac{1}{\alpha}(I-C)x$. The projected gradient method on Ψ_α with step-size α then takes the form:

$$\begin{aligned} x^{(k+1)} &= \mathcal{P}_{\mathcal{X}^N}\left\{x^{(k)} - \alpha\nabla\Psi_\alpha(x^{(k)})\right\} \quad (9)\\ &= \mathcal{P}_{\mathcal{X}^N}\left\{x^{(k)} - \right.\\ &\qquad \left. \alpha\left(\nabla F(x^{(k)}) + \frac{1}{\alpha}(I-C)x^{(k)}\right)\right\},\end{aligned}$$

which, after rearranging terms, is precisely (4).

It is easy to see that Ψ_α is strongly convex on $\mathbb{R}^N$, with modulus $\mu' = \mu$ (which equals the strong convexity modulus of the f_i's). Further, $\nabla\Psi_\alpha$ is Lipschitz continuous on $\mathbb{R}^N$, with constant $L' = L + \frac{1-\lambda_N(C)}{\alpha}$. Namely, $\forall x, y \in \mathbb{R}^N$:

$$\begin{aligned}\|\nabla\Psi_\alpha(x) - \nabla\Psi_\alpha(y)\| &\leq \|\nabla F(x) - \nabla F(y)\| \\ &+ \frac{1}{\alpha}\|I-C\|\,\|x-y\| \\ &\leq L\|x-y\| + \frac{1-\lambda_N(C)}{\alpha}\|x-y\|.\end{aligned}$$

[3]The point of convergence of algorithm (4) and the solution to (8) are intentionally denoted by the same symbol because – as we will show – they actually are the same point.

(Note that $\|\nabla F(x) - \nabla F(y)\| \leq L\|x-y\|$ follows after summing the inequalities: $|\nabla f_i(x_i) - \nabla f_i(y_i)|^2 \leq L^2|x_i - y_i|^2$, $i=1,...,N$, and using $\|\nabla F(x)\|^2 = \sum_{i=1}^N |\nabla f_i(x_i)|^2$.) We impose that α satisfies $\alpha \leq \frac{1}{L'}$, which, after simple manipulations, gives: $\alpha \leq \lambda_N(C)/(L)$, as introduced before.

An immediate consequence of the fact that algorithm (4) is precisely the projected gradient method to solve (8) is the following Lemma, first observed in [46].

Lemma 6 ([46]) Standard distributed gradient algorithm (4) with step-size $\alpha \leq \lambda_N(C)/L$ converges to the point $x^\bullet \in \mathcal{X}^N$ – the solution to (8).

We proceed by proving Theorem 1.

Proof of Theorem 1: As per Lemma 6, algorithm (4) converges to $x^\bullet$ – the solution to (8). We hence need to prove for the solution to (8) the characterization in (5).

Consider an arbitrary point $x \in \mathcal{X}^N$, and let $\overline{x} := \frac{1}{N}\sum_{i=1}^N x_i$. We first prove the following inequality:

$$f(\overline{x}) - f(x^\star) \leq (\Psi_\alpha(x) - \Psi_\alpha(x^\bullet)) + \frac{\alpha N G^2}{2(1-\lambda_2(C))}. \quad (10)$$

Indeed, we have that:

$$\begin{aligned} x^\top(I-C)x &= (x - \overline{x}\mathbf{1})^\top (I-C)(x-\overline{x}\mathbf{1}) \\ &\geq \lambda_{N-1}(I-C)\|x - \overline{x}\mathbf{1}\|^2 \\ &= (1-\lambda_2(C))\|\widetilde{x}\|^2, \end{aligned}$$

where we let $\widetilde{x} := x - \overline{x}\mathbf{1}$. Further,

$$\begin{aligned} \sum_{i=1}^N f_i(x_i) &= \sum_{i=1} f_i(\overline{x}) + (\sum_{i=1}(f_i(x_i) - f_i(\overline{x}))) \\ &\geq f(\overline{x}) - G\sum_{i=1}^N |x_i - \overline{x}| \\ &\geq f(\overline{x}) - G\sqrt{N}\|\widetilde{x}\|. \end{aligned}$$

The second from last inequality follows because $f_i(x_i) \geq f_i(\overline{x}) + \nabla f_i(\overline{x})(x_i - \overline{x}) \geq f_i(\overline{x}) - G\,|x_i - \overline{x}|$. Combining the previous conclusions:

$$\begin{aligned} \Psi_\alpha(x) - \Psi_\alpha(x^\bullet) &\geq f(\overline{x}) - \Psi_\alpha(x^\bullet) - G\sqrt{N}\|\widetilde{x}\| \\ &+ \frac{1}{2\,\alpha}(1-\lambda_2(C))\|\widetilde{x}\|^2 \\ &\geq f(\overline{x}) - \Psi_\alpha(x^\bullet) \\ &- \sup_{t\geq 0}\left\{G\sqrt{N}t - \frac{1}{2\alpha}(1-\lambda_2(C))t^2\right\} \\ &\geq f(\overline{x}) - \Psi_\alpha(x^\bullet) - \frac{\alpha\, N\, G^2}{2(1-\lambda_2(C))}. \quad (11) \end{aligned}$$

Next, note that $\Psi_\alpha(x^\bullet) = \min_{x\in\mathcal{X}^N}\Psi_\alpha(x) \leq \Psi_\alpha(x^\star\mathbf{1}) = f(x^\star)$, and so $-\Psi_\alpha(x^\bullet) \geq -f(x^\star)$. Applying this to (11), completes the proof of (10).

We now prove claim (5) in Theorem 1. We have:

$$\begin{aligned} \|x^\bullet - x^\star\mathbf{1}\|^2 &= \|x^\bullet - \overline{x}^\bullet\mathbf{1} + \overline{x}^\bullet\mathbf{1} - x^\star\mathbf{1}\|^2 \\ &\leq 2\,\|x^\bullet - \overline{x}^\bullet\mathbf{1}\|^2 + 2N\,|\overline{x}^\bullet - x^\star|^2. \quad (12) \end{aligned}$$

For the second summand in (12), we have:

$$\begin{aligned} |\overline{x}^\bullet - x^\star|^2 &\leq \frac{2}{\mu}(f(\overline{x}^\bullet) - f(x^\star)) \\ &\leq \frac{\alpha N G^2}{\mu(1-\lambda_2(C))}. \end{aligned}$$

The first inequality above is due to strong convexity of f, and the second applies (10) with $x = x^\bullet$ (where $\overline{x}^\bullet = \frac{1}{N}\sum_{i=1}^N x_i^\bullet$). We now upper bound the first summand in (12). We have that:

$$\begin{aligned} \Psi_\alpha(x^\bullet) &= \sum_{i=1}^N f_i(x_i^\bullet) + \frac{1}{2\alpha}(x^\bullet)^\top(I-C)x^\bullet \\ &\geq \frac{1-\lambda_2(C)}{2\alpha}\|\widetilde{x}^\bullet\|^2 + Nm_f, \end{aligned}$$

where $\widetilde{x}^\bullet = x^\bullet - \overline{x}^\bullet\mathbf{1}$. On the other hand,

$$\Psi_\alpha(x^\bullet) \leq f(x^\star) \leq N\,M_f.$$

Combining the obtained upper and lower bounds on $\Psi_\alpha(x^\bullet)$, we obtain for the first summand in (12):

$$\|x^\bullet - \overline{x}^\bullet\mathbf{1}\|^2 \leq \frac{2\alpha N(M_f - m_f)}{1-\lambda_2(C)}.$$

Combining the bounds on the first and second summands, the claim in (5) follows.

It remains to prove the claim in (7). By standard analysis of gradient methods, we have that:

$$\|x^{(k)} - x^\bullet\| \leq (1-\alpha\mu)^k\|x^{(0)} - x^\bullet\| \leq (1-\alpha\mu)^k 2\sqrt{N}D,$$

where we used that $\|x^{(0)}\| \leq \sqrt{N}D$, and the same bound for $x^\bullet$. Thus, the desired result. ■

C. *Analysis of the proposed method* (2)

We now turn our attention to the proposed method (3). It is easy to verify that (3) can be written as:

$$x^{(k+1)} = \mathcal{P}_{\mathcal{X}^N}\left\{x^{(k)} - \alpha\left[\nabla\Psi_\alpha(x^{(k)}) + e^{(k)}\right]\right\}, \quad (13)$$

where $e^{(k)} = (e_1^{(k)},...,e_N^{(k)})^\top$ is a random vector, with i-th component equal to:

$$\begin{aligned} e_i^{(k)} &= \left(\frac{z_i^{(k)}}{p_k} - 1\right)\nabla f_i(x_i^{(k)}) \\ &+ \frac{1}{\alpha}\sum_{j\in\Omega_i} C_{ij}\,(z_i^{(k)}z_j^{(k)} - 1)\left(x_i^{(k)} - x_j^{(k)}\right). \quad (14) \end{aligned}$$

Hence, (2) is an inexact projected gradient method applied to Ψ_α, with step-size α, where the amount of inexactness is given by vector $e^{(k)}$.

Overall, our strategy in analyzing (13) consists of two main steps: 1) analyzing the inexact projected gradient method (13); and 2) characterizing (upper bounding) the inexactness vector $e^{(k)}$. For the former step, we apply Proposition 3 in [23]. Adapted to our setting, the proposition says the following. Consider minimization of $\phi(y)$ over $y \in \mathcal{Y}$, where $\phi : \mathbb{R}^m \to \mathbb{R}$ is a convex function, and $\mathcal{Y} \subset \mathbb{R}^m$ is a closed convex set. Let $y^\bullet$ be the solution to the above problem.

Further, let ϕ be strongly convex with modulus $\mu_\phi > 0$, and let ϕ have a Lipschitz continuous gradient with constant $L_\phi \geq \mu_\phi$.

Lemma 7 (Proposition 3, [23]) Consider the algorithm:

$$y^{(k+1)} = \mathcal{P}_{\mathcal{Y}}\left\{y^{(k)} - \frac{1}{L_\phi}\left[\nabla\phi(y^{(k)}) + e_y^{(k)}\right]\right\},\ k = 0, 1, ...,$$

where $e_y^{(k)}$ is a random vector. Then, $\forall k = 1, 2, ...$:

$$\begin{aligned}\|y^{(k)} - y^\bullet\| &\leq (1 - \mu_\phi/L_\phi)^k \|y^{(0)} - y^\bullet\| \\ &+ \frac{1}{L_\phi}\sum_{t=1}^{k}(1 - \mu_\phi/L_\phi)^{k-t}\|e_y^{(t-1)}\|, \quad (15)\end{aligned}$$

where $y^{(0)} \in \mathcal{Y}$ is the initial point.

Note that, if $\nabla\phi$ is Lipschitz continuous with constant L_ϕ, then $\nabla\phi$ is also Lipschitz continuous with constant $1/\alpha \geq L_\phi$. Therefore, for the function ϕ and the iterations:

$$y^{(k+1)} = \mathcal{P}_{\mathcal{Y}}\left\{y^{(k)} - \alpha\left[\nabla\phi(y^{(k)}) + e_y^{(k)}\right]\right\},\ k = 0, 1, ...,$$

there holds:

$$\begin{aligned}\|y^{(k)} - y^\bullet\| &\leq (1 - \alpha\,\mu_\phi)^k \|y^{(0)} - y^\bullet\| \\ &+ \alpha\sum_{t=1}^{k}(1 - \alpha\,\mu_\phi)^{k-t}\|e_y^{(t-1)}\|,\ k = 1, ..(16)\end{aligned}$$

In other words, the modified claim (16) holds even if we take a step size different (smaller than) $1/L_\phi$.

For analyzing the inexact projected gradient method (13), we will also make use of the following result. (The first claim of it is Lemma 3.1 in in [12].)

Lemma 8 (Lemma 3.1, [12]) Consider a deterministic sequence $\{v_k\}$ such that $v_k \to 0$ as $k \to \infty$, and let a be a constant in $(0, 1)$. Then, there holds:

$$\sum_{t=1}^{k} a^{k-t} v_{t-1} \to 0. \quad (17)$$

If, moreover, there exist positive constants $\mathcal{C}_v$ and ζ such that, for all $k = 0, 1, ...,$

$$v_k \leq \frac{\mathcal{C}_v}{(k+1)^{1+\zeta}},$$

then there exists positive constant $\mathcal{C}_v'$ such that, for all $k = 1, 2, ...,$

$$\sum_{t=1}^{k} a^{k-t} v_{t-1} \leq \frac{\mathcal{C}_v'}{k^{1+\zeta}}. \quad (18)$$

Step 1: gradient inexactness. We proceed by characterizing the gradient inexactness; Lemma 9 upper bounds quantity $\mathbb{E}\left[\|e^{(k)}\|^2\right]$.

Lemma 9 (Gradient inexactness) For all $k = 0, 1, ...,$ there holds:

$$\begin{aligned}\mathbb{E}\left[\|e^{(k)}\|^2\right] &\leq 4(1 - p_k)\frac{N\,G^2}{p_{\min}} \\ &+ 72(1 - p_k^2)\frac{NG^2}{(p_{\min})^2(1-\beta)^2} \\ &\leq \mathcal{C}_e\,(1 - p_k^2), \quad (19)\end{aligned}$$

where

$$\mathcal{C}_e = \frac{4\,N\,G^2}{p_{\min}} + \frac{72\,NG^2}{(p_{\min})^2(1-\beta)^2}. \quad (20)$$

Proof: Consider (14). We have:

$$\begin{aligned}|e_i^{(k)}|^2 &\leq 2\left|\frac{z_i^{(k)}}{p_k} - 1\right|^2 |\nabla f_i(x_i^{(k)})|^2 \quad (21) \\ &+ \frac{2}{\alpha^2}\sum_{j\in\Omega_i} C_{ij}\,|z_i^{(k)} z_j^{(k)} - 1|^2 \left|x_i^{(k)} - x_j^{(k)}\right|^2 \\ &\leq 2G^2\left|\frac{z_i^{(k)}}{p_k} - 1\right|^2 + \frac{4}{\alpha^2}\sum_{j\in\Omega_i} C_{ij}\,|z_i^{(k)} z_j^{(k)} - 1|^2 \\ &\times \left(\left|\widetilde{x}_i^{(k)}\right|^2 + \left|\widetilde{x}_j^{(k)}\right|^2\right). \quad (22)\end{aligned}$$

Inequality (21) uses the following bound: $(u+v)^2 \leq 2u^2 + 2v^2$. It also uses, with $u_i := (z_i^{(k)} z_j^{(k)} - 1)(x_i^{(k)} - x_j^{(k)})$, the following relation:

$$\begin{aligned}(\sum_{j\in\Omega_i} C_{ij}u_j)^2 &= (\sum_{j\in\Omega_i} C_{ij}u_j + C_{ii}\cdot 0)^2 \\ &\leq \sum_{j\in\Omega_i} C_{ij}u_j^2 + C_{ii}\cdot 0^2 \\ &= \sum_{j\in\Omega_i} C_{ij}u_j^2,\end{aligned}$$

which follows due to the fact that $\sum_{j\in\Omega_i} C_{ij}u_j + C_{ii}\cdot 0$ is a convex combination, and $v \mapsto v^2$, $v \in \mathbb{R}$, is convex. Inequality (22) uses that

$$\begin{aligned}\left|x_i^{(k)} - x_j^{(k)}\right|^2 &= \left|x_i^{(k)} - \overline{x}^{(k)} + \overline{x}^{(k)} - x_j^{(k)}\right|^2 \\ &\leq 2\left|x_i^{(k)} - \overline{x}^{(k)}\right|^2 + 2\left|\overline{x}^{(k)} - x_j^{(k)}\right|^2.\end{aligned}$$

Taking expectation, and using independence of $x^{(k)}$ from $z^{(k)}$:

$$\begin{aligned}\mathbb{E}\left[|e_i^{(k)}|^2\right] &\leq 2G^2\,\mathbb{E}\left[\left|\frac{z_i^{(k)}}{p_k} - 1\right|^2\right] \\ &+ \frac{4}{\alpha^2}\sum_{j\in\Omega_i} C_{ij}\,\mathbb{E}\left[|z_i^{(k)} z_j^{(k)} - 1|^2\right] \\ &\times \left(\mathbb{E}\left[\left|\widetilde{x}_i^{(k)}\right|^2\right] + \mathbb{E}\left[\left|\widetilde{x}_j^{(k)}\right|^2\right]\right). \quad (23)\end{aligned}$$

We proceed by upper bounding $\mathbb{E}\left[\left|\frac{z_i^{(k)}}{p_k} - 1\right|^2\right]$, using the total probability law with respect to the following partition:

$\{z_i^{(k)}=1\}$, and $\{z_i^{(k)}=0\}$:

$$\mathbb{E}\left[\left|\frac{z_i^{(k)}}{p_k}-1\right|^2\right]=\left|\frac{1}{p_k}-1\right|^2\mathbb{P}(z_i^{(k)}=1)+P\left(z_i^{(k)}=0\right)$$

$$\begin{aligned} &= \left|\frac{1}{p_k}-1\right|^2 p_k+(1-p_k) && (24)\\ &= \frac{1}{p_k}(1-p_k)^2+(1-p_k)\\ &\le \frac{1}{p_k}(1-p_k)+(1-p_k)\\ &\le 2(1-p_k)/p_{\min}. && (25)\end{aligned}$$

We next upper bound $\mathbb{E}\left[|z_i^{(k)}z_j^{(k)}-1|^2\right]$, using the total probability law with respect to the event $\{z_i^{(k)}=1,\, z_j^{(k)}=1\}$ and its complement; we obtain:

$$\begin{aligned}\mathbb{E}\left[|z_i^{(k)}z_j^{(k)}-1|^2\right] &= (1-\mathbb{P}(z_i^{(k)}=1,\, z_j^{(k)}=1))\\ &= (1-p_k)^2. && (26)\end{aligned}$$

Substituting (25) and (26) in (23):

$$\begin{aligned}\mathbb{E}\left[|e_i^{(k)}|^2\right] &\le 4\,G^2\,(1-p_k)/p_{\min}\\ &+ \frac{4}{\alpha^2}\sum_{j\in\Omega_i}C_{ij}\,(1-p_k^2)\\ &\times \left(\mathbb{E}\left[\left|\widetilde{x}_i^{(k)}\right|^2\right]+\mathbb{E}\left[\left|\widetilde{x}_j^{(k)}\right|^2\right]\right). && (27)\end{aligned}$$

Summing the above inequalities over $i=1,...,N$, using the fact that $\sum_{j\in\Omega_i}C_{ij}\le 1$, $\forall i$, $\mathbb{E}\left[\|e^{(k)}\|^2\right]=\sum_{i=1}^N\mathbb{E}\left[|e_i^{(k)}|^2\right]$, and $\mathbb{E}\left[\|\widetilde{x}^{(k)}\|^2\right]=\sum_{i=1}^N\mathbb{E}\left[|\widetilde{x}_i^{(k)}|^2\right]$, we obtain:

$$\begin{aligned}\mathbb{E}\left[\|e^{(k)}\|^2\right] &\le 4\,N\,G^2\,(1-p_k)/p_{\min}\\ &+ \frac{8}{\alpha^2}\,(1-p_k^2)\,\mathbb{E}\left[\left\|\widetilde{x}^{(k)}\right\|^2\right].\end{aligned}$$

Finally, applying Lemma 5 to the last inequality, the claim follows. ■

Step 2: Analyzing the inexact projected gradient method. We first state and prove the following Lemma on algorithm (2).

Lemma 10 Consider algorithm (2) with step-size $\alpha \le \lambda_N(C)/(L)$. Then, for the iterates $x^{(k)}$ and $x^\bullet$–the solution to (8), $\forall k=1,2,...$, there holds:

$$\begin{aligned}\mathbb{E}\left[\|x^{(k)}-x^\bullet\|^2\right] &\le 8\,N\,(1-\alpha\mu)^{2k}D^2\\ &+ \frac{\alpha\,\mathcal{C}_e}{\mu}\sum_{t=1}^k(1-\alpha\,\mu)^{k-t}(1-p_{t-1}^2).\end{aligned}$$

Proof: As already established, algorithm (2) is an inexact projected gradient method to solve (8), with the inexactness vector $e^{(k)}$. We now apply (16) to sequence $x^{(k)}$ and iterations (3); we obtain:

$$\begin{aligned}\|x^{(k)}-x^\bullet\| &\le (1-\alpha\,\mu)^k\|x^{(0)}-x^\bullet\|\\ &+ \alpha\sum_{t=1}^k(1-\alpha\mu)^{k-t}\|e^{(t-1)}\|. && (28)\end{aligned}$$

Squaring the latter inequality, using $(u+v)^2\le 2u^2+2v^2$, and $\|x^{(0)}-x^\bullet\|\le 2\sqrt{N}D$:

$$\begin{aligned}\|x^{(k)}-x^\bullet\|^2 &\le 8(1-\alpha\,\mu)^{2k}ND^2\\ &+ \alpha^2\left(\sum_{t=0}^k(1-\alpha\mu)^{k-t}\right)\\ &\times \sum_{t=1}^k(1-\alpha\mu)^{k-t}\|e^{(t-1)}\|^2. && (29)\end{aligned}$$

In (29), we used the following. Let $\theta_t=(1-\alpha\mu)^{k-t}$, and $S_t:=\sum_{t=1}^k\theta_t$. Then,

$$\begin{aligned}\left(\sum_{t=1}^k\theta_t\|e^{(t-1)}\|\right)^2 &= S_t^2\left(\sum_{t=1}^k\frac{\theta_t}{S_t}\|e^{(t-1)}\|\right)^2\\ &\le S_t^2\sum_{t=1}^k\frac{\theta_t}{S_t}\|e^{(t-1)}\|^2\\ &= S_t\sum_{t=1}^k\theta_t\|e^{(t-1)}\|^2,\end{aligned}$$

where we used convexity of the scalar quadratic function $v\mapsto v^2$. Now, using $\sum_{t=1}^k(1-\alpha\mu)^{k-t}\le\frac{1}{1-(1-\alpha\mu)}=\frac{1}{\alpha\mu}$, (29) is further upper bounded as:

$$\begin{aligned}\|x^{(k)}-x^\bullet\|^2 &\le 8(1-\alpha\,\mu)^{2k}ND^2\\ &+ \frac{\alpha^2}{\alpha\mu}\sum_{t=1}^k(1-\alpha\mu)^{k-t}\|e^{(t-1)}\|^2.\end{aligned}$$

Taking expectation, and applying Lemma 9, we obtain the claimed result. ■

We are now ready to prove Theorems 2 and 3.

Proof of Theorem 2: The proof of the mean square sense convergence claim follows from Lemma 10 by applying (17). Namely, setting $a:=1-\alpha\mu$ and $v_t:=1-p_t^2$, the desired result follows.

We now prove the almost sure convergence claim. By Lemma 10, using $p_k=1-u_k$, we have:

$$\begin{aligned}\mathbb{E}\left[\|x^{(k)}-x^\bullet\|^2\right] &\le 8\,N\,(1-\alpha\mu)^{2k}D^2\\ &+ \frac{2\alpha\,\mathcal{C}_e}{\mu}\sum_{t=1}^k(1-\alpha\,\mu)^{k-t}u_{t-1}.\end{aligned}$$

Now, from (18), there exists a positive constant $\mathcal{C}_u'$ such that, for all $k=1,2,...$:

$$\sum_{t=1}^k(1-\alpha\,\mu)^{k-t}u_{t-1}\le\frac{\mathcal{C}_u'}{k^{1+\zeta}},$$

and hence:

$$\mathbb{E}\left[\|x^{(k)}-x^\bullet\|^2\right] \leq 8\,N\,(1-\alpha\mu)^{2k}D^2 + \frac{2\alpha\,\mathcal{C}_e\,\mathcal{C}_u'}{\mu}\frac{1}{k^{1+\zeta}}.$$

Summing the above inequality over $k=1,2,...$, we obtain that:

$$\sum_{k=0}^{\infty}\mathbb{E}\left[\|x^{(k)}-x^\bullet\|^2\right] < \infty. \tag{30}$$

Applying the Chebyshev's inequality and (30), we conclude that:

$$\sum_{k=0}^{\infty}\mathbb{P}\left(\|x^{(k)}-x^\bullet\| > \epsilon\right) < \infty,$$

for any $\epsilon > 0$. Therefore, by the first Borel-Cantelli lemma, $\mathbb{P}\left(\|x^{(k)}-x^\bullet\| > \epsilon,\ \text{infinitely often}\right) = 0$, which finally implies that $x^{(k)}$ converges to $x^\bullet$, almost surely. ■

Proof of Theorem 3: Consider (28). Taking expectation:

$$\begin{aligned}\mathbb{E}\left[\|x^{(k)}-x^\bullet\|\right] &\leq \sqrt{N}(1-\alpha\,\mu)^k 2D \qquad (31)\\ &+ \alpha\sum_{t=1}^{k}(1-\alpha\mu)^{k-t}\sqrt{\mathcal{C}_e(1-p_{t-1}^2)}\\ &\leq \sqrt{N}(1-\alpha\,\mu)^k 2D \qquad (32)\\ &+ \alpha\sum_{t=1}^{k}(1-\alpha\mu)^{k-t}\sqrt{\mathcal{C}_e}\sqrt{2}(\sqrt{\delta})^t.\end{aligned}$$

The first inequality uses $\mathbb{E}[|u|] \leq (\mathbb{E}[|u|^2])^{1/2}$. The second inequality uses $1-p_{t-1}^2 = (1-(1-\delta^t))^2 \leq 2\delta^t$. Consider the sum in (32). For each t, each summand is upper bounded by $\eta^k\sqrt{2\mathcal{C}_e}$, and so the sum is $O(k\eta^k)$. The term $\sqrt{N}(1-\alpha\,\mu)^k 2D = O(\eta^k)$. Hence, the overall right-hand-side in (32) is $O(k\eta^k) = O((\eta+\epsilon)^k)$, which completes the proof. ■

Remark 5 The proof of Theorem 3 also determines the constant in the convergence rate. From the above proof, substituting the expression for $\mathcal{C}_e$ in (20), it is straightforward to observe that, for all $k=1,2,...$:

$$\mathbb{E}\left[\|x^{(k)}-x^\bullet\|\right] \leq 12\,\max\left\{\sqrt{N}\,D,\ \frac{\alpha\sqrt{N}\,G}{p_{\min}(1-\beta)}\right\}k\,\eta^k.$$

V. Simulations

We provide simulations on the problem of learning a linear classifier via logistic loss, both on synthetic and real data sets. Simulations demonstrate that our proposed idling strategy significantly reduces the total cost (both communication and computational costs), when compared with standard distributed gradient method where all nodes work at all iterations. At the same time, the proposed method does not increase – or even reduces – the total number of iterations. Simulations also demonstrate the method's high degree of robustness to asynchrony and its benefits over gossip-based strategies for solving (1).

We consider distributed learning of a linear classifier via logistic loss, e.g., [47]. Each node i possesses J data samples $\{a_{ij}, b_{ij}\}_{j=1}^J$. Here, $a_{ij} \in \mathbb{R}^3$ is a feature vector, and $b_{ij} \in \{-1,+1\}$ is its class label. We want to learn a vector $x = (x_1^\top, x_0)^\top$, $x_1 \in \mathbb{R}^{d-1}$, and $x_0 \in \mathbb{R}$, $d \geq 1$, such that the corresponding linear classifier $\operatorname{sign}(H_x(a)) = \operatorname{sign}\left(x_1^\top a + x_0\right)$ minimizes the total surrogate loss with l_2 regularization:

$$\sum_{i=1}^{N}\left(\sum_{j=1}^{J}\mathcal{J}_{\text{logis}}\left(b_{ij}H_x(a_{ij})\right) + \frac{1}{2}\mathcal{R}\|x\|^2\right), \tag{33}$$

subject to a prior knowledge that $\|x\| \leq \mathcal{M}$, where $\mathcal{M} > 0$ is a constant. Here, $\mathcal{J}_{\text{logis}}(\cdot)$ is the logistic loss $\mathcal{J}_{\text{logis}}(\alpha) = \log(1+e^{-\alpha})$, and $\mathcal{R}$ is a positive regularization parameter. Clearly, problem (33) fits the generic framework in (1) with $f_i(x) = \sum_{j=1}^J \mathcal{J}_{\text{logis}}\left(b_{ij}H_x(a_{ij})\right) + \mathcal{R}\|x\|^2$, $f(x) = \sum_{i=1}^N f_i(x)$, and $\mathcal{X} = \{x \in \mathbb{R}^4 : \|x\| \leq \mathcal{M}\}$. A strong convexity constant of the f_i's μ can be taken as $\mu = \frac{\mathcal{R}}{N}$, while a Lipschitz constant L can be taken as $\frac{1}{4N}\|\sum_{i=1}^N\sum_{j=1}^J c_{ij}\,c_{ij}^\top\| + \frac{\mathcal{R}}{N}$, where $c_{ij} = (b_{ij}\,a_{ij}^\top, b_{ij})^\top$.

With all experiments, we test the algorithms on a connected network with $N=50$ nodes and 214 links, generated as a random geometric graph: we place nodes randomly (uniformly) on a unit square, and the node pairs whose distance is less than a radius are connected by an edge.

Experiments on synthetic data. In the first set of experiments, we generate data and set the algorithm parameters as follows. Each node i has $J=2$ data points whose dimension is $d-1=3$. We generate the a_{ij}'s independently over i and j; each entry of a_{ij} is drawn independently from the standard normal distribution. We generate the "true" vector $x^\star = ((x_1^\star)^\top, x_0^\star)^\top$ by drawing its entries independently from standard normal distribution. Then, the class labels are generated as $b_{ij} = \operatorname{sign}\left((x_1^\star)^\top a_{ij} + x_0^\star + \epsilon_{ij}\right)$, where ϵ_{ij}'s are drawn independently from normal distribution with zero mean and standard deviation 0.1. The obtained corresponding strong convexity parameter $\mu = 0.1$, and the Lipschitz constant $L \approx 0.69$. Further, we set $\mathcal{M} = 100$ and $\mathcal{R} = 0.1$.

With both algorithms, we initialize $x_i(0)$ to $\mathcal{P}_{\mathcal{X}}(h_i)$, where the h_i's, $i=1,...,N$, are generated mutually independently, and the entries of each h_i are generated mutually independently from the uniform distribution on $[-50,+50]$. $\forall i$. We utilize the Metropolis weights, e.g., [48]. With the proposed method, we set $p_k = 1-\delta^{k+1}$, $k=0,1,...$, and $\delta = (1-\alpha\,\mu)^2$.

As an error metric, we use the relative error in the objective function averaged across nodes:

$$\frac{1}{N}\sum_{i=1}^{N}\frac{f(x_i^{(k)})-f^\star}{f^\star},\ f^\star > 0,$$

where $f^\star$ is evaluated numerically via the (centralized) projected gradient method. With the proposed method, we run 200 simulations and consider both the average relative error (averaged across 200 simulation runs with different instantiations of node activations, i.e., variables $z(k)$) and the relative error's histograms.

We compare the two methods with respect to the total cost (total number of activations across all nodes), where a unit cost corresponds to a single node activation at one iteration; we also include the comparisons with respect to the number of iterations k. We consider two different values of step-sizes, $\alpha \in \{\frac{1}{250\,L}, \frac{1}{50\,L}\}$, which correspond to different achievable accuracies by both methods.

Figure 1 compares the proposed and standard distributed gradient methods for $\alpha = 1/(50L)$. We can see that the proposed method significantly reduces the total cost to reach a certain accuracy, while at the same time it does not induce overhead in the total number of iterations. For example, we can see from Figure 1 (a) that, to achieve relative error $\epsilon = 0.01$, the proposed method has on average the total cost around $16,700$, while the standard distributed gradient method requires around $25,000$, which gives relative savings of about 33%. At the same time, the two methods require practically the same number of iterations (Figure 1 (b)). Figure 1 (c) shows the histogram for the proposed method of the total cost to achieve relative error $\epsilon = 0.01$, where an arrow indicates the cost of the standard distributed gradient method. Figure 1 (d) repeats the study for the total number of iterations. Figure **??** shows the comparisons for $\alpha = 1/(250L)$, and it shows histograms to reach $\epsilon = 0.005$, again demonstrating very significant improvements.

Experiments on real world data sets. In the second set of experiments, we consider the same network with 50 nodes and test the algorithms on a real world data sets, "a1a," which we downloaded from the repository: http://www.csie.ntu.edu.tw/ cjlin/libsvm/. With data set "a1a," we have $N\,J = 1,600$ data points ($J = 32$ per node) of dimension $d - 1 = 119$ (optimization variable dimension is 120); with "Mushrooms," $N\,J = 8,100$ ($J = 162$) and $d - 1 = 112$. we set $\mathcal{M} = 100$ and $\mathcal{R} = 0.1$.

We use all the system and algorithmic parameters the same as in the first set of experiments, except the following. With the proposed method, we set $p_k = \max\{1 - \delta^{k+1}, 0.1\}$, $k = 0, 1, \ldots$, and $\delta = \min\{(1 - \alpha\,\mu)^2, 0.99999\}$. (See the discussion in*****.) As error metrics, we use the average cost function (averaged across nodes)

$$\frac{1}{N}\sum_{i=1}^{n} f\left(x_i^{(k)}\right),$$

and the normalized gradient norm:

$$\frac{\frac{1}{N}\sum_{i=1}^{N}\left\|\nabla f\left(x_i^{(k)}\right)\right\|}{\frac{1}{N}\sum_{i=1}^{N}\left\|\nabla f\left(x_i^{(0)}\right)\right\|},$$

where $\sum_{i=1}^{N}\left\|\nabla f\left(x_i^{(0)}\right)\right\| > 0$. [4] With the proposed method, we run one sample path realization.

Figure 3 compares the proposed and standard distributed gradient methods for "a1a" data set, for step size $\alpha = 1/(50L)$ We can see that the proposed method reduces the total cost by at an order of magnitude, and it also significantly reduces the number of iterations for convergence.

Modeling and testing asynchronous operation. In applications like, e.g., WSNs, accounting for asynchrony in the algorithm's operation is highly relevant. In such scenarios, when node i decides to activate at iteration k and transmit a message to node j, this message may be lost, due to, e.g., packet dropouts in WSNs. In addition, an active node may fail to calculate the local gradient at iteration k, because the actual calculation may take longer than the time slot allocated to iteration k, or due to unavailability of sufficient computational resources at k. Therefore, under asynchrony, the schedule of realized inter-neighbor communications and local gradient evaluations is not under the full control of networked nodes.

We introduce the following model. At each link $\{i, j\} \in E$ and each $k = 0, 1, \ldots$, let $\widehat{z}_{\{i,j\}}^{(k)}$ be a binary random variable which takes value one if the communication link is online and zero otherwise; let $\widehat{p}_{ij} := \mathbb{P}\left(\widehat{z}_{\{i,j\}}^{(k)} = 1\right)$. Therefore, variable $\widehat{z}_{\{i,j\}}^{(k)}$ models a failure of link $\{i, j\}$ at k. Similarly, for each node i, introduce a binary random variable $\widehat{z}_i^{(k)}$, which takes the value one if the calculation of $\nabla f_i\left(x_i^{(k)}\right)$ is successful and zero otherwise. We let $\widehat{p}_i := \mathbb{P}\left(\widehat{z}_i^{(k)} = 1\right)$. Variable $\widehat{z}_{\{i,j\}}^{(k)}$ hence models failure of node i's gradient calculation at k. (As before, each node i activates if $z_i^{(k)} = 1$ and stays idle if $z_i^{(k)} = 0$.) We assume that the variables $\widehat{z}_{\{i,j\}}^{(k)}$ are independent both across links and across iterations; likewise, the $\widehat{z}_i^{(k)}$'s are independent both across nodes and across iterations; and that the node activations, the link failures, and the gradient calculation failures are mutually independent processes. Note that $z_i^{(k)}$ is in control of node i, while the $\widehat{z}_{\{i,j\}}^{(k)}$'s and $\widehat{z}_i^{(k)}$'s are governed "by nature." The update of node i and iteration k is as follows. If $z_i^{(k)} = 0$, node i stays idle; else, if $z_i^{(k)} = 1$, we have:

$$\begin{aligned} x_i^{(k+1)} &= \mathcal{P}_{\mathcal{X}}\{\,(1 - \sum_{j \in \Omega_i} z_i^{(k)}\widehat{z}_{\{i,j\}}^{(k)} C_{ij}\,)x_i^{(k)} \\ &+ \sum_{j \in \Omega_i} C_{ij}\, z_i^{(k)}\widehat{z}_{\{i,j\}}^{(k)}\, x_j^{(k)} \\ &- \frac{\alpha}{p_k} z_i^{(k)}\widehat{z}_i^{(k)}\nabla f_i(x_i^{(k)})\,\}. \end{aligned} \tag{34}$$

Note that we assume that nodes do not have prior knowledge on the asynchrony parameters $\widehat{p}_i$'s and $\widehat{p}_{\{i,j\}}$'s.

We provide a simulation example on the synthetic data set and the 50-node network considered before with the same simulation parameters and $\alpha = 1/(50L)$. Each $\widehat{p}_{\{i,j\}}$ is set to 0.5, while for the $\widehat{p}_i$'s we consider two scenarios: 1) lower failure probabilities, where half of the nodes have $\widehat{p}_i = 0.9$ and the other half has $\widehat{p}_i = 0.5$; and 2) higher failure probabilities, where half of the nodes have $\widehat{p}_i = 0.9$ and the other half has $\widehat{p}_i = 0.1$. Note that the latter scenario corresponds to rather severe conditions, as half of the nodes successfully computes gradients only with 0.1 probability.

Figure 4 shows the performance of the proposed method

[4]This is a valid performance metric, because we numerically verified that, with this example, solution $x^\star$ is not at the boundary of the constraint set, i.e., we have that $\|\nabla f(x^\star)\| = 0$. We verified this by establishing that $f(0) \leq f(x)$, for all boundary points x, i.e., for all x with $\|x\| = \mathcal{M}$, and hence the solution is not at the boundary of the constraint set. This claim was easy to check by numerically evaluating $f(0)$ and numerically verifying that $f(0) \leq \frac{N\mathcal{R}}{2}\mathcal{M}^2$, which implies, due to strong convexity of f (with modulus $N\mathcal{R}$), that $f(0) \leq f(x)$, for all x with $\|x\| = \mathcal{M}$.

for three scenarios: no failures, lower failure probabilities, and higher failure probabilities. We can see that the proposed algorithm exhibits a very strong resilience to asynchrony. First consider the higher failure probabilities scenario (dashed line in Figure 4 (a)). We can see that, despite the severe conditions, the proposed algorithm still converges close to the solution, naturally with a decreased convergence rate and with a moderately increased limiting error. Now, consider the lower failure probabilities scenario (dotted line in Figure 4 (a)). The proposed algorithm again generally slows down convergence, as it is expected. However, interestingly, it actually achieves a higher degree of accuracy asymptotically than under the synchronous scenario. This is explained as follows. The effective step-size of node i with algorithm (34) equals $\frac{\alpha}{p_k} z_i^{(k)} \widehat{z}_i^{(k)}$, which is on average $\alpha \widehat{p}_i$. Hence, in a sense, $\widehat{p}_i$ has the effect of decreasing the step-size. The step-size decrease has the known effect of slowing down convergence rate but improving the asymptotic accuracy, as confirmed in Figure 4 (a). The improved asymptotic accuracy indeed occurs as long as the $\widehat{p}_i$'s are not mutually too different. When the $\widehat{p}_i$'s are mutually too far apart, different nodes effectively use very different step-sizes (which equal to $\alpha \widehat{p}_i$), and this disbalance makes a negative effect on both the convergence speed and on the asymptotic accuracy – as confirmed in Figure 4 (a) for the higher failure probabilities case.

Comparison with a gossip-based scheme. To further corroborate the benefits of the proposed idling scheme with increasing activation probabilities, we compare it on the synthetic data set and $\alpha = 1/(50L)$ with the gossip-based scheme in [13]. We can see that the proposed scheme outperforms gossip. Most notably, the gossip scheme has a larger steady state error under the fair – equal step sizes – comparison. We explain why this happens. Namely, with gossip, only two nodes (out of N) are active at all iterations. This means that, essentially, the gossip-based scheme behaves as an incremental gradient method (more precisely, a mini-batch) gradient method, where the full gradient (which equals the sum of N local nodes functions' gradients) is at all times approximated with the sum of two local gradients. Therefore, the gossip-based scheme incurs an increased steady state error, for a similar reason as the fact why the (centralized) incremental gradient method with constant step size does not converge to the exact solution. In contrast, our method essentially behaves as a full gradient method, thus leading to a higher accuracy.

VI. Discussion and extensions

A. *Relaxing strong convexity and differentiability*

We assumed throughout the previous part of the paper that the f_i's are strongly convex and have Lipschitz continuous gradients. We now extend our results to more generic cost functions, when these two assumptions are relaxed. Specifically, we now let each $f_i : \mathbb{R}^d \to \mathbb{R}$ be convex and Lipschitz over set $\mathcal{X}$, i.e.,

$$|f_i(x) - f_i(y)| \le G\,\|x - y\|,\ \forall x, y \in \mathcal{X}. \tag{35}$$

We continue to assume that $\mathcal{X}$ is convex and compact, so the class of the f_i's which satisfies (35) is fairly wide.

The proposed algorithm (2) generalizes straightforwardly: at node i and iteration k, gradient $\nabla f_i\left(x_i^{(k)}\right)$ is replaced with an arbitrary subgradient from subdifferential set $\partial f_i\left(x_i^{(k)}\right)$. We note that Lemmas 5 and 9 continue to hold here as well.

Before presenting our result on the modified algorithm (2), we recall that the standard distributed gradient method achieves for the setting assumed here the following performance. Define, for each node i, the running average:

$$x_{i,\mathrm{ra}}^{(k)} = \frac{1}{k}\sum_{t=0}^{k-1} x_i^{(t)},\ k = 1, 2, ...$$

Then, for all i (see, e.g., [44]):

$$f\left(x_{i,\mathrm{ra}}^{(k)}\right) - f^\star \le O\left(\frac{1}{\alpha\, k}\right) + O\left(\alpha\right). \tag{36}$$

For method (2), we show the following. Assume that activation probability $p_k = 1 - u_k$, $u_k \ge 0$, $\forall k$, satisfies that:

$$S_u := \sum_{k=0}^{\infty} \sqrt{u_k} < \infty. \tag{37}$$

Then, for all i, for all $k = 1, 2, ...$:

$$\begin{aligned} & \mathbb{E}\left[f(\overline{x}_{\mathrm{ra}}^{(K)}) - f^\star\right] \qquad (38)\\ \le\ & \frac{4ND^2}{2\alpha K} + \frac{4\sqrt{2}\alpha\,\sqrt{N}\,D\,\sqrt{\mathcal{C}_e}\,S_u}{K} \\ +\ & 2\alpha^2 G_\Psi^2 + 4\alpha^2\mathcal{C}_e + \frac{\alpha N G^2}{2(1-\lambda_2(C))} + \frac{3\alpha N G^2}{p_{\min}(1-\beta)}. \end{aligned}$$

Therefore, as long as p_k converges to one sufficiently fast (per condition (37) it suffices to have, e.g., $p_k = 1 - \frac{1}{(k+1)^{2+\zeta}}$, $\zeta > 0$ arbitrarily small), the idling schedule does not violate the $O\left(\alpha + \frac{1}{\alpha\, k}\right)$ bound.

B. *Quantifying reduction in the total cost*

Although Theorem 2 demonstrates that the proposed method achieves practically the same convergence factor (in terms of iterations k) as the standard distributed gradient method, the Theorem does not explicitly quantify the cost reduction needed for achieving a prescribed ϵ-accuracy. Quantifying this in full generality is very challenging. We pursue here the special case of quadratic costs with identity Hessians.

Setting. We let $f_i : \mathbb{R}^d \to \mathbb{R}$ be $f_i(x) = \frac{1}{2}\|x - b_i\|^2$, $i = 1, ..., N$, where the b_i's are constant vectors in $\mathbb{R}^d$. Note that $\mu = L = 1$ and $x^\star = \frac{1}{N}\sum_{i=1}^N b_i$. Denote by $b^\star := \mathbf{1} \otimes x^\star$ and $b := (b_1, ..., b_N)^\top \in \mathbb{R}^{Nd}$. For simplicity, we consider equal weights $C_{ij} = c_0$, for all $\{i, j\} \in E$, so that weight matrix $C = I - c_0\,\mathcal{L}$, where $\mathcal{L}$ is the (un-normalized) zero-one graph Laplacian matrix. Then, for $c_0 \le \lambda_N(\mathcal{L})$, we have $\|W - J\| = 1 - c_0\,\lambda_2(\mathcal{L})$. From now on, we write simply $\lambda_i = \lambda_i(\mathcal{L})$. Denote by $R_{\mathrm{sp}} := \|[\,(I - J) \otimes I\,]\,b\|$, and by $R_0 := \|x^{(0)} - b^\star\|$. Quantity R_{sp} measures how spread are the b_i's, i.e., how the b_i's (minimizers of the individual f_i's) are far apart from solution $x^\star = \frac{1}{N}\sum_{i=1}^N b_i$. With the proposed method, we set $p_k = 1 - \frac{1}{2}\delta^{k+1}$, $k = 0, 1, ...$, with $p_k = 1 - \alpha\,\theta$, $\theta \in (0, 1/\alpha]$. (This is a slightly different choice from one considered in the rest of the paper.) We consider as error metric the norm of the mean distance to the solution: $\left\|\mathbb{E}\left[x^{(k)}\right] - b^\star\right\|$. This is of course not a very strong metric, but nonetheless it allows to derive neat expressions. We denote the latter quantity with the standard distributed gradient method by $\xi^{(k)}$ and with the proposed method by $\chi^{(k)}$.

Intermediate results. We derive the following upper bounds on $\xi^{(k)}$ and $\chi^{(k)}$, respectively. For all $k = 1, 2, ...$, there holds:

$$\begin{aligned} \xi^{(k)} &\leq \xi_{\mathrm{ub}}^{(k)} := (1-\alpha)^k R_0 + \alpha R_{\mathrm{sp}}(N-1) \\ &\times \frac{1-(1-\alpha-c_0\lambda_2)^k}{c_0\lambda_2+\alpha} \end{aligned} \tag{39}$$

$$\begin{aligned} \chi^{(k)} &\leq \chi_{\mathrm{ub}}^{(k)} := (1-\alpha)^k R_0 + \alpha R_{\mathrm{sp}}(N-1) \\ &\times \left(\frac{1}{c_0\lambda_2(1-\delta^{k/2})+\alpha} + \frac{(1-\alpha-c_0\lambda_2(1-\delta))^{(k-1)/2}}{c_0\lambda_2(1-\delta)+\alpha} \right). \end{aligned} \tag{40}$$

Results. Based on the above inequalities, we derive the following result. Let the desired accuracy be ϵ, i.e., we want that: $\xi_{\mathrm{ub}}^{(k)} \leq \epsilon$ and $\chi_{\mathrm{ub}}^{(k)} \leq \epsilon$. Then, for $\alpha = \frac{c_0\lambda_2\epsilon}{2(N-1)R_{\mathrm{sp}}}$ and $\theta > \frac{1}{c_0\lambda_2}$, after:

$$K_\epsilon = \frac{R_{\mathrm{sp}}(N-1)}{c_0\lambda_2\epsilon} 2\ln\left(\frac{2R_0}{\epsilon}\right)$$

iterations, we have that

$$\xi_{\mathrm{ub}}^{(k)} = \epsilon(1+o(\epsilon)) \text{ and } \chi_{\mathrm{ub}}^{(k)} = \epsilon(1+o(\epsilon)),$$

i.e., both algorithms achieve the same error ϵ after the same number of iterations K_ϵ (up to lower orders in ϵ). Therefore, the proposed method achieves savings in total cost (per node) equal to:

$$K_\epsilon - \sum_{k=0}^{K_\epsilon} p_k,$$

which is approximately

$$\frac{1}{2\alpha\theta} = \frac{(N-1)R_{\mathrm{sp}}}{c_0\lambda_2\epsilon}\frac{1}{\theta}.$$

VII. Conclusion

We explored the effect of two sources of redundancy with distributed projected gradient algorithms. The first redundancy, well-known in the literature on distributed multi-agent optimization, stems from the fact that not all inter-neighbor links need to be utilized at all iterations for the algorithm to converge. The second redundancy, explored before only in centralized optimization, arises when we minimize the sum of cost functions, each summand corresponding to a distinct data sample. In this setting, it is known that performing a gradient method with an appropriately increasing sample size can exhibit convergence properties that essentially match the properties of a standard gradient method, where the full sample size is utilized at all times. We simultaneously explored the two sources of redundancy for the first time to develop a novel distributed gradient method. With the proposed method, each node, at each iteration k, is active with a certain probability p_k, and is idle with probability $1 - p_k$, where the activation schedule is independent across nodes and across iterations. Assuming that the nodes' local costs are strongly convex and have Lipschitz continuous gradients, we showed that the proposed method essentially matches the linear convergence rate (towards a solution neighborhood) of the standard distributed projected gradient method, where all nodes are active at all iterations. Simulations on l_2-regularized logistic losses demonstrate that the proposed method significantly reduces the total communication and computational cost to achieve a desired accuracy, when compared with the standard distributed gradient method. As a future work, we plan to apply the proposed idling nodes strategy to other distributed multi-agent algorithms, including, e.g., the Nesterov gradient variants.

Appendix

A. *Proof of result (38) in Subsection VI-A*

Assume for simplicity that $d = 1$ but the proof extends to a generic $d > 1$. Let $x^\bullet$ be a solution of (8) (which exists as $\Psi_\alpha : \mathbb{R}^N \to \mathbb{R}$ is continuous and constraint set $\mathcal{X}$ is compact.) The update of the proposed method can be written as:

$$x^{(k+1)} = \mathcal{P}_{\mathcal{X}^N}\left\{x^{(k)} - \alpha\left[g_\Psi^{(k)} + e^{(k)}\right]\right\}.$$

Here, $g_\Psi^{(k)}$ is a subgradient of Ψ_α at $x^{(k)}$ which equals:

$$\begin{aligned} g_\Psi^{(k)} &= g_F^{(k)} + \frac{1}{\alpha}(I-C)x^{(k)} \\ &= g_F^{(k)} + \frac{1}{\alpha}(I-C)\widetilde{x}^{(k)}, \end{aligned}$$

where $g_F^{(k)} = (g_1^{(k)}, ..., g_N^{(k)})^\top$, and $g_i^{(k)}$ is a subgradient of f_i at $x_i^{(k)}$. Also, recall $e^{(k)}$ from (14). Note that Lemmas 5 and 9 continue to hold here as well, and therefore, it is easy to show that, for all $k = 0, 1, ...$:

$$\mathbb{E}\left[\|g_\Psi^{(k)}\|^2\right] \leq G_\Psi^2 := 2NG^2 + \frac{18NG^2}{(p_{\min})(1-\beta)^2}. \tag{41}$$

Now, a standard analysis of projected subgradient methods, following, e.g., [49], gives:

$$\begin{aligned} \|x^{(k+1)} - x^\bullet\|^2 &\leq \|x^{(k)} - x^\bullet\|^2 \\ &- 2\alpha\left(x^{(k)} - x^\bullet\right)^\top (g_\Psi^{(k)} + e^{(k)}) \\ &+ \alpha^2\|g_\Psi^{(k)} + e^{(k)}\|^2. \end{aligned}$$

Using $\left\|x^{(k)} - x^\bullet\right\| \leq 2\sqrt{N}D$, and

$$\Psi_\alpha(x^\bullet) \geq \Psi_\alpha(x^{(k)}) + (g_\Psi^{(k)})^\top(x^\bullet - x^{(k)}),$$

we further obtain:

$$\begin{aligned} &\|x^{(k+1)} - x^\bullet\|^2 \leq \|x^{(k)} - x^\bullet\|^2 - 2\alpha(\Psi_\alpha(x^{(k)}) \\ &-\Psi_\alpha(x^\bullet)) + 4\alpha\sqrt{N}D\|e^{(k)}\| + 2\alpha^2\|g_\Psi^{(k)}\|^2 + 2\alpha^2\|e^{(k)}\|^2. \end{aligned}$$

Summing the above inequality for $k = 0, ..., K-1$, dividing the resulting inequality by K, and using (41), we obtain:

$$\frac{2\alpha}{K}\sum_{k=0}^{K-1}\left(\Psi_\alpha(x^{(k)}) - \Psi_\alpha(x^\bullet)\right)$$
$$\leq \frac{\|x^{(0)} - x^\bullet\|^2}{K} + \frac{4\alpha\sqrt{N}D}{K}\sum_{k=0}^{K-1}\|e^{(k)}\|$$
$$+ \frac{2\alpha^2}{K}\sum_{k=0}^{K-1}\|g_\Psi^{(k)}\|^2 + \frac{2\alpha}{K}\sum_{k=0}^{K-1}\|e^{(k)}\|^2.$$

Consider the running average $x_{\text{ra}}^{(K)} := \frac{1}{K}\sum_{k=0}^{K-1} x^{(k)}$. Using convexity of Ψ_α, applying (41), and taking expectation:

$$\mathbb{E}\left[\Psi_\alpha(x^{(K)}) - \Psi_\alpha(x^\bullet)\right] \tag{42}$$
$$\leq \frac{4ND^2}{2\alpha K} + \frac{4\alpha\sqrt{N}D}{K}\sum_{k=0}^{K-1}\mathbb{E}[\|e^{(k)}\|]$$
$$+ 2\alpha^2 G_\Psi^2 + \frac{2\alpha^2}{K}\sum_{k=0}^{K-1}\mathbb{E}[\|e^{(k)}\|^2].$$

Next, note that $\mathbb{E}[\|e^{(k)}\|] \leq \sqrt{2\mathcal{C}_e}\sqrt{u_k}$, and $\mathbb{E}[\|e^{(k)}\|^2] \leq 2\mathcal{C}_e u_k$, where we recall $p_k = 1 - u_k$. Applying the latter bounds on (42), Using the facts that $u_k \geq 1$, for all k, and that $S_u := \sum_{k=0}^{\infty}\sqrt{u_k}$, we obtain:

$$\mathbb{E}\left[\Psi_\alpha(x_{\text{ra}}^{(K)}) - \Psi_\alpha(x^\bullet)\right] \tag{43}$$
$$\leq \frac{4ND^2}{2\alpha K} + \frac{4\sqrt{2}\alpha\sqrt{N}\,D\,\sqrt{\mathcal{C}_e}\,S_u}{K}$$
$$+ 2\alpha^2 G_\Psi^2 + 4\alpha^2\mathcal{C}_e.$$

Applying (10) to $x_{\text{ra}}^{(k)}$. defining $\overline{x}_{\text{ra}}^{(k)} := \frac{1}{N}\sum_{i=1}^{N} x_{i,\text{ra}}^{(k)}$, and taking expectations, it follows that:

$$\mathbb{E}\left[f(\overline{x}_{\text{ra}}^{(K)}) - f^\star\right] \tag{44}$$
$$\leq \frac{4ND^2}{2\alpha K} + \frac{4\sqrt{2}\alpha\sqrt{N}\,D\,\sqrt{\mathcal{C}_e}\,S_u}{K}$$
$$+ 2\alpha^2 G_\Psi^2 + 4\alpha^2\mathcal{C}_e + \frac{\alpha N G^2}{2(1-\lambda_2(C))}.$$

Finally, using the same argument as in [18], equation (22), we obtain the desired result.

B. Proof of results in Subsection VI-B

We let $d = 1$ for notational simplicity. Consider $x^{(k)} - b^\star$, where we recall $b^\star = \frac{1}{N}\sum_{i=1}^{N} b_i\,\mathbf{1}$. Then, it is easy to show that, for $k = 0, 1, ...$, the following recursive equation holds:

$$\left(x^{(k+1)} - b^\star\right) = \widetilde{C}\left(x^{(k)} - b^\star\right) + \alpha(I - J)b, \tag{45}$$

where $\widetilde{C} = C - \alpha I$. Therefore, for $k = 1, 2, ...$, we have:

$$x^{(k)} - b^\star = \widetilde{C}^k(x^{(0)} - b^\star) + \alpha\sum_{t=0}^{k-1}\widetilde{C}^{k-t}(I - J)b. \tag{46}$$

Let q_i denote the i-th unit-norm eigenvector, and λ_i the i-th eigenvalue of Laplacian $\mathcal{L}$, ordered in an ascending order. We have that $\lambda_1 = 0$, $\lambda_2 > 0$, and $q_1 = \frac{1}{\sqrt{N}}\mathbf{1}$. Further, note that $\|\widetilde{C}\| = 1 - \alpha$. Then, there holds:

$$\widetilde{C}^{k-t}(I - J)b = \sum_{i=2}^{N}(1 - c_0\lambda_i - \alpha)^{k-t}\widetilde{q}_i\widetilde{q}_i^\top(I - J)b,$$

because $q_1^\top(I - J)b = 0$. Therefore, from (46), we obtain:

$$\xi^{(k)} \leq (1-\alpha)^k R_0 + \alpha R_{\text{sp}}(N-1)\sum_{t=0}^{k-1}(1 - \alpha - c_0\lambda_2)^{k-t},$$

which yields (39).

Now, we consider algorithm (2). Recall quantity $\chi^{(k)} = \|\mathbb{E}[x^{(k)}] - b^\star\|$. Considering the recursive equation on $\mathbb{E}[x^{(k)}] - b^\star$, completely analogously to the above, we can obtain:

$$\chi^{(k)} \leq (1-\alpha)^k R_0 \tag{47}$$
$$+ \alpha R_{\text{sp}}(N-1)\sum_{t=0}^{k-1}\prod_{s=t}^{k-1}\left(1 - \alpha - c_0\lambda_2(1 - \delta^{s+1})\right).$$

We now upper bound the sum in (47). We split the sum into two parts:

$$\mathcal{S}_1 = \sum_{t=0}^{(k-1)/2}\prod_{s=t}^{k-1}\left(1 - \alpha - c_0\lambda_2(1 - \delta^{s+1})\right) \tag{48}$$
$$\mathcal{S}_2 = \sum_{t=(k-1)/2+1}^{k-1}\prod_{s=t}^{k-1}\left(1 - \alpha - c_0\lambda_2(1 - \delta^{s+1})\right). \tag{49}$$

(To avoid notational complications, we consider even k and $k \geq 2$.) Next, note that

$$\mathcal{S}_1 \leq \sum_{t=0}^{(k-1)/2}\left(1 - \alpha - c_0\lambda_2(1 - \delta)\right)^{k-1-t}$$
$$\mathcal{S}_2 \leq \sum_{t=(k-1)/2+1}^{k-1}\left(1 - \alpha - c_0\lambda_2(1 - \delta^{k/2})\right)^{k-1-t}.$$

From the above bounds, it is easy to show that (40) follows.

Now, let $K_\epsilon = \frac{\ln\left(\frac{2R_0}{\epsilon}\right)}{\alpha}$, and $\alpha = \frac{c_0\lambda_2\epsilon}{2R_{\text{sp}}(N-1)}$. It is easy to show that, for these values, quantity $\xi_{\text{ub}}^{(k)}$ in (39) is $\epsilon(1 + o(\epsilon))$. We now show that quantity $\chi_{\text{ub}}^{(k)}$ in (40) is also $\epsilon(1 + o(\epsilon))$. First, note that $(1-\alpha)^{K_\epsilon} = \frac{\epsilon}{2}(1 + o(\epsilon))$. Next, consider the term:

$$\frac{\alpha R_{\text{sp}}(N-1)}{c_0\lambda_2(1 - \delta^{k/2}) + \alpha} \leq \frac{\alpha R_{\text{sp}}(N-1)}{c_0\lambda_2(1 - \delta^{k/2})}.$$

Note that, for $\theta > 0$, we have that $\delta^{k/2} \sim \left(\frac{\epsilon}{2R_0}\right)^{\theta/2} = o(1)$. Therefore, we have that:

$$\frac{\alpha R_{\text{sp}}(N-1)}{c_0\lambda_2(1 - \delta^{k/2})} = \frac{\epsilon}{2}(1 + o(1)).$$

Therefore, we must show that the remaining term:

$$\frac{\alpha R_{\text{sp}}(N-1)\left(1 - \alpha - c_0\,\lambda_2(1-\delta)\right)^{(k-1)/2}}{c_0\lambda_2(1-\delta) + \alpha}$$
$$\leq R_{\text{sp}}(N-1)\left(1 - \alpha - c_0\,\lambda_2(1-\delta)\right)^{(k-1)/2} = o(\epsilon).$$

Observe that:

$$(1-\alpha-c_0\,\lambda_2(1-\delta))^{(k-1)/2}\sim\left(\frac{\epsilon}{2R_0}\right)^{(1+c_0\lambda_2\theta)/2}.$$

This term is $o(1)$ is if $\theta>1/(c_0\lambda_2)$, which we assumed, and therefore we conclude that $\chi_{\text{ub}}^{(k)}$ in (40) is $\epsilon(1+o(1))$.

References

[1] A. Daneshmand, F. Facchinei, V. Kungurtsev, and G. Scutari, "Hybrid random/deterministic parallel algorithms for nonconvex big data optimization," *IEEE Transactions on Signal Processing*, vol. 63, no. 15, pp. 3914–3929, Aug. 2015.

[2] K. Slavakis, G. B. Giannakis, and G. Mateos, "Modeling and optimization for big data analytics," *IEEE Signal Processing Magazine*, vol. 31, pp. 18–31, 2014.

[3] K. Slavakis, S.-J. Kim, G. Mateos, and G. B. Giannakis, "Stochastic approximation vis-a-vis online learning for big data analytics," *IEEE Signal Processing Magazine*, vol. 31, no. 11, pp. 124–129, Nov. 2014.

[4] I. D. Schizas, A. Ribeiro, and G. B. Giannakis, "Consensus in ad hoc WSNs with noisy links – Part I: Distributed estimation of deterministic signals," *IEEE Trans. Sig. Process.*, vol. 56, no. 1, pp. 350–364, Jan. 2009.

[5] M. Rabbat and R. Nowak, "Distributed optimization in sensor networks," in *IPSN 2004, 3rd International Symposium on Information Processing in Sensor Networks*, Berkeley, California, USA, April 2004, pp. 20 – 27.

[6] S. Kar, J. M. F. Moura, and K. Ramanan, "Distributed parameter estimation in sensor networks: Nonlinear observation models and imperfect communication," *IEEE Transactions on Information Theory*, vol. 58, no. 6, pp. 3575–3605, June 2012.

[7] C. Lopes and A. H. Sayed, "Adaptive estimation algorithms over distributed networks," in *21st IEICE Signal Processing Symposium*, Kyoto, Japan, Nov. 2006.

[8] F. Cattivelli and A. H. Sayed, "Diffusion LMS strategies for distributed estimation," *IEEE Trans. Sig. Process.*, vol. 58, no. 3, pp. 1035–1048, March 2010.

[9] I. Necoara and J. A. K. Suykens, "Application of a smoothing technique to decomposition in convex optimization," *IEEE Trans. Autom. Contr.*, vol. 53, no. 11, pp. 2674–2679, Dec. 2008.

[10] J. Mota, J. Xavier, P. Aguiar, and M. Püschel, "Distributed optimization with local domains: Applications in mpc and network flows," *IEEE Trans. Autom. Contr.*, vol. 60, no. 7, pp. 2004–2009, July 2015.

[11] A. Nedic and A. Ozdaglar, "Distributed subgradient methods for multi-agent optimization," *IEEE Transactions on Automatic Control*, vol. 54, no. 1, pp. 48–61, January 2009.

[12] S. Ram, A. Nedic, and V. Veeravalli, "Distributed stochastic subgradient projection algorithms for convex optimization," *Jour. Opt. Theory and App.*, vol. 147, no. 3, pp. 516–545, 2011.

[13] S. S. Ram, A. Nedic, and V. Veeravalli, "Asynchronous gossip algorithms for stochastic optimization," in *CDC '09, 48th IEEE International Conference on Decision and Control*, Shanghai, China, December 2009, pp. 3581 – 3586.

[14] A. Mokhtari, Q. Ling, and A. Ribeiro, "Network Newton-part I: Algorithm and convergence," *submitted to IEEE Transactions on Signal Processing*, 2015, available at: http://arxiv.org/abs/1504.06017.

[15] ——, "Network Newton-part II: Convergence rate and implementation," *IEEE Transactions on Signal Processing*, 2015, available at: http://arxiv.org/abs/1504.06020.

[16] ——, "Network newton," in *Asilomar Conference on signals, systems, and computers*, Pacific Grove, CA, November 2014, pp. 1621–1625.

[17] M. Zargham, A. Ribeiro, A. Ozdaglar, and A. Jadbabaie, "Accelerated dual descent for network flow optimization," *IEEE Transactions on Automatic Control*, vol. 59, no. 4, pp. 905–920, 2014.

[18] D. Jakovetic, J. Xavier, and J. M. F. Moura, "Fast distributed gradient methods," *IEEE Trans. Autom. Contr.*, vol. 59, no. 5, pp. 1131–1146, May 2014.

[19] I.-A. Chen and A. Ozdaglar, "A fast distributed proximal gradient method," in *Allerton Conference on Communication, Control and Computing*, Monticello, IL, October 2012, pp. 601–608.

[20] K. Yuan, Q. Ling, and W. Yin, "On the convergence of decentralized gradient descent," *to appear in SIAM Journal on Optimization*, 2015, available at: http://arxiv.org/abs/1310.7063.

[21] I. Matei and J. S. Baras, "Performance evaluation of the consensus-based distributed subgradient method under random communication topologies," *IEEE Journal of Selected Topics in Signal Processing*, vol. 5, no. 4, pp. 754–771, 2011.

[22] I. Lobel and A. Ozdaglar, "Convergence analysis of distributed subgradient methods over random networks," in *46th Annual Allerton Conference onCommunication, Control, and Computing*, Monticello, Illinois, September 2008, pp. 353 – 360.

[23] M. Schmidt, N. L. Roux, and F. Bach, "Convergence rates of inexact proximal-gradient methods for convex optimization," in *Advances in Neural Information Processing Systems 24*, 2011, pp. 1458–1466.

[24] J. Tsitsiklis, D. Bertsekas, and M. Athans, "Distributed asynchronous deterministic and stochastic gradient optimization algorithms," *IEEE Trans. Autom. Contr.*, vol. 31, no. 9, pp. 803–812, Sep. 1986.

[25] S. Kar and J. M. F. Moura, "Sensor networks with random links: Topology design for distributed consensus," *IEEE Transactions on Signal Processing*, vol. 56, no. 7, pp. 3315–3326, July 2008.

[26] T. C. Aysal, A. D. Sarwate, and A. G. Dimakis, "Reaching consensus in wireless networks with probabilistic broadcast," in *47th Annual Allerton Conference on Communication, Control, and Computing*, Monticello, IL, Oct. 2009, pp. 732–739.

[27] T. Aysal, M. Yildiz, A. Sarwate, and A. Scaglione, "Broadcast gossip algorithms for consensus," *IEEE Transactions on Signal Processing*, vol. 57, no. 7, pp. 2748–2761, July 2009.

[28] S. Kar and J. M. F. Moura, "Distributed consensus algorithms in sensor networks: Quantized data and random link failures," *IEEE Trans. Sig. Process.*, vol. 58, no. 3, pp. 1383–1400, March 2010.

[29] N. Takahashi and I. Yamada, "Link probability control for probabilistic diffusion least-mean squares over resource-constrained networks," in *ICASSP 2010, IEEE Int. Conf. on Acoustics, Speech, and Signal Processing*, Dallas, TX, March 2010, pp. 3518–3521.

[30] X. Zhao and A. Sayed, "Asynchronous adaptation and learning over networks-part I: Modeling and stability analysis," *IEEE Transactions on Signal Processing*, vol. 63, no. 4, pp. 811–826, Feb. 2015.

[31] ——, "Asynchronous adaptation and learning over networks-part II: Performance analysis," *IEEE Transactions on Signal Processing*, vol. 63, no. 4, pp. 827–842, Feb. 2015.

[32] ——, "Asynchronous adaptation and learning over networks-part III: Comparison analysis," *IEEE Transactions on Signal Processing*, vol. 63, no. 4, pp. 843–858, Feb. 2015.

[33] S.-Y. Tu and A. H. Sayed, "On the influence of informed agents on learning and adaptation over networks," *IEEE Trans. Signal Processing*, vol. 61, no. 6, pp. 1339–1356, March 2013.

[34] K. I. Tsianos, S. F. Lawlor, J. Y. Yu, and M. G. Rabbat, "Networked optimization with adaptive communication," in *IEEE GlobalSIP Network Theory Sumposium*, Austin, Texas, December 2013, pp. 579–582.

[35] G. Deng and M. C. Ferris, "Variable-number sample path optimization," *Mathematical Programming*, vol. 117, no. 1–2, pp. 81–109, 2009.

[36] T. H. de Mello, "Variable-sample methods for stochastic optimization," *ACM Transactions on Modeling and Computer Simulation*, vol. 13, no. 2, pp. 108–133, 2003.

[37] E. Polak and J. O. Royset, "Efficient sample sizes in stochastic nonlinear programing," *Journal of Computational and Applied Mathematics*, vol. 217, no. 2.

[38] R. Pasupathy, "On choosing parameters in restrospective-approximation algorithms for simulation-optimization," in *2006 Winter Simulation Conference, L.F. Perrone, F.P. Wieland, J. Liu, B.G. Lawson, D.M. Nicol and R.M. Fujimoto, eds.*, 2006, pp. 208–215.

[39] ——, "On choosing parameters in retrospective-approximation algorithms for stochastic root finding and simulation optimization," *Operations Research*, vol. 58, no. 4, pp. 889–901, 2010.

[40] F. Bastin, "Trust-region algorithms for nonlinear stochastic programming and mixed logit models," 2004, phD Thesis, University of Namur, Belgium.

[41] F. Bastin, C. Cirillo, and P. L. Toint, "An adaptive monte carlo algorithm for computing mixed logit estimators," *Computational Management Science*, vol. 3, no. 1, pp. 55–79, 2006.

[42] N. Krejić and N. Krklec, "Line search methods with variable sample size for unconstrained optimization," *Journal of Computational and Applied Mathematics*, vol. 245, pp. 213–231, 2013.

[43] N. Krejić and N. K. Jerinkić, "Nonmonotone line search methods with variable sample size," *Numerical Algorithms*, vol. 68, no. 4, pp. 711–739, 2015, DOI: 10.1007/s11075-014-9869-1.

[44] A. Nedic, A. Ozdaglar, and A. Parrilo, "Constrained consensus and optimization in multi-agent networks," *IEEE Transactions on Automatic Control*, vol. 55, no. 4, pp. 922–938, April 2010.

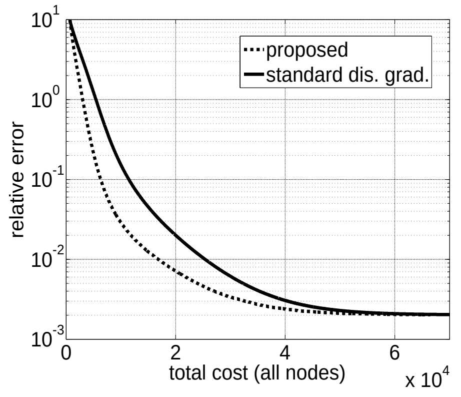
[45] D. Bajovic, J. Xavier, J. M. F. Moura, and B. Sinopoli, "Consensus and products of random stochastic matrices: Exact rate for convergence in probability," *IEEE Trans. Sig. Process.*, vol. 61, no. 10, pp. 2557–2571, May 2013.

[46] D. Jakovetic, J. M. F. Moura, and J. Xavier, "Distributed Nesterov-like gradient algorithms," in *CDC'12, 51st IEEE Conference on Decision and Control*, Maui, Hawaii, December 2012, pp. 5459–5464.

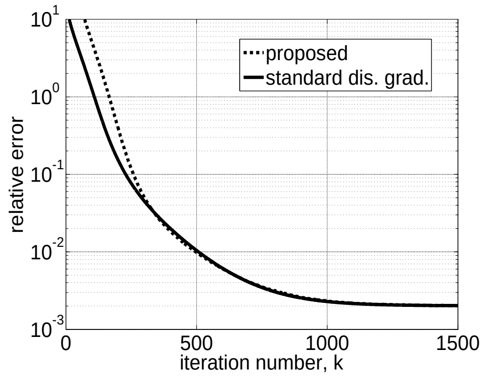
[47] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, "Distributed optimization and statistical learning via the alternating direction method of multipliers," *Foundations and Trends in Machine Learning, Michael Jordan, Editor in Chief*, vol. 3, no. 1, pp. 1–122, 2011.

[48] L. Xiao, S. Boyd, and S. Lall, "A scheme for robust distributed sensor fusion based on average consensus," in *IPSN '05, Information Processing in Sensor Networks*, Los Angeles, California, 2005, pp. 63–70.

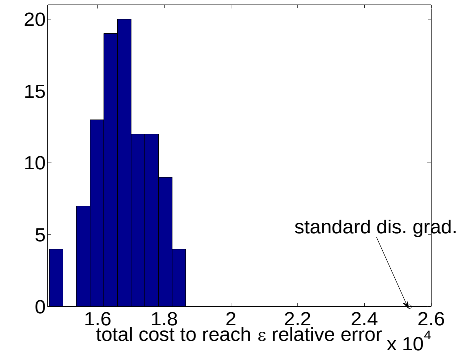
[49] A. Nedic and D. Bertsekas, "The effect of deterministic noise in subgradient methods," *Mathematical Programming*, vol. 125, no. 1, pp. 75–99, Jan. 2010.



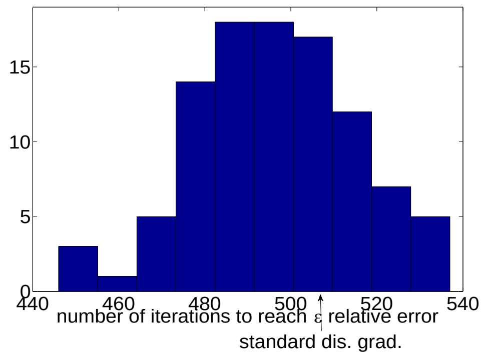
(a) Average relative error vs. total cost (all nodes).



(b) Average relative error vs. number of iterations.

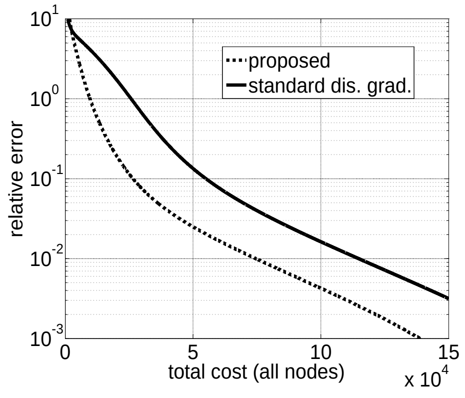


(c) Histogram: total cost to reach rel. err. 0.01.

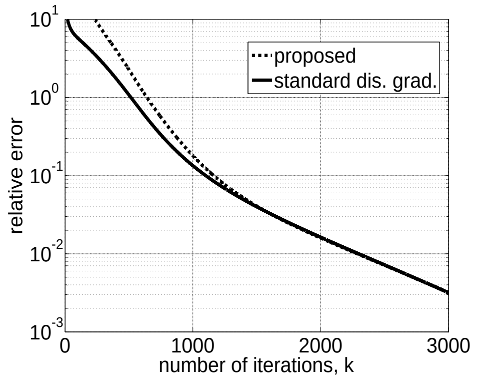


(d) Histogram: #iterations to reach rel. err. 0.01.

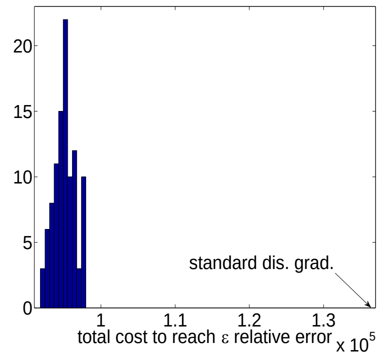
Fig. 1: Comparison of the proposed and standard distributed gradient methods for $\alpha = \frac{1}{50\,L}$.



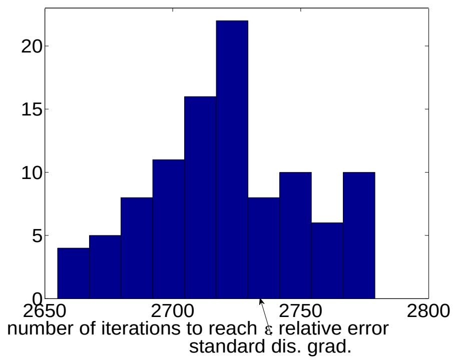
(a) Average relative error vs. total cost (all nodes).



(b) Average relative error vs. number of iterations.

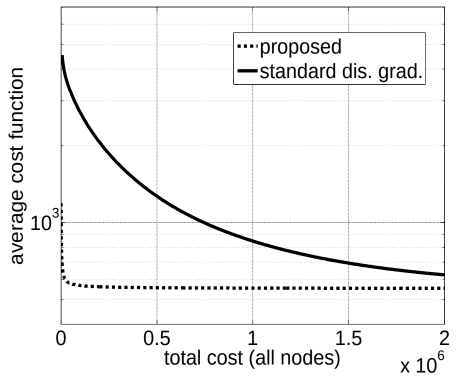


(c) Histogram: total cost to reach rel. err. 0.005.

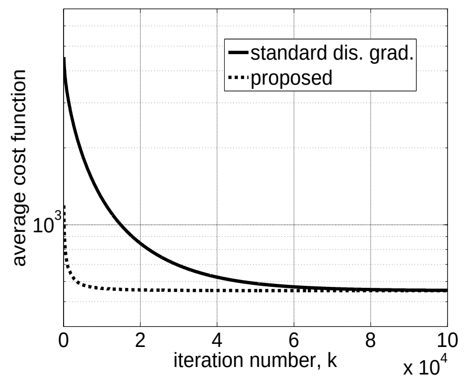


(d) Histogram: total cost to reach rel. err. 0.005.

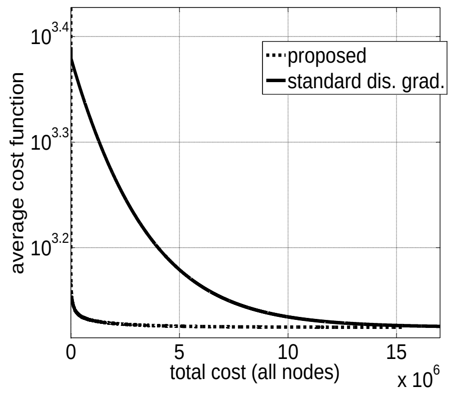
Fig. 2: Comparison of the proposed and standard distributed gradient methods for $\alpha = \frac{1}{250\,L}$.



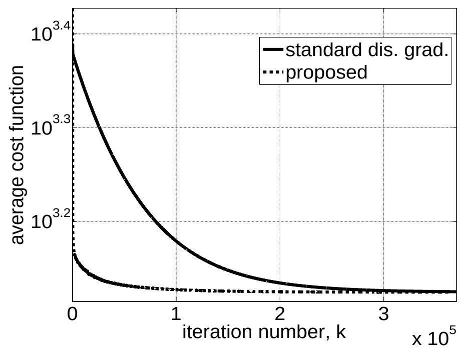
(a) Average cost function vs. total cost for “a1a.”



(b) Average cost function vs. number of iterations for “a1a.”

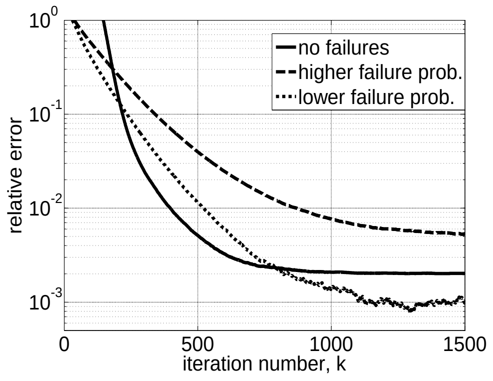


(c) Average cost function vs. total cost for “Madelon.”

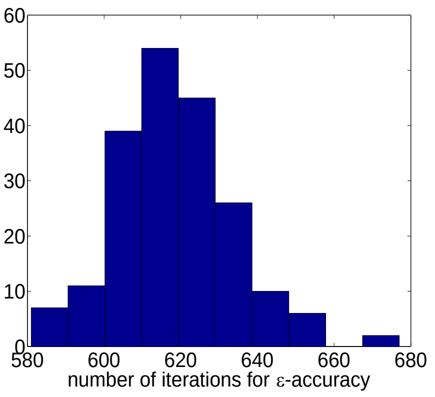


(d) Average cost function vs. number of iterations for “Madelon.”

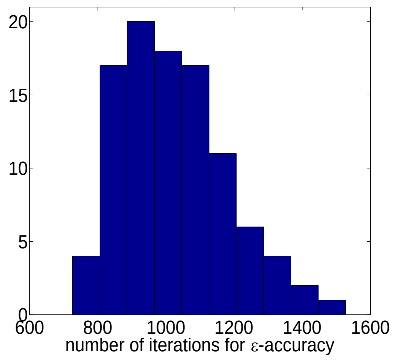
Fig. 3: Comparison of the proposed and standard distributed gradient methods for data sets “a1a” and “Madelon.”



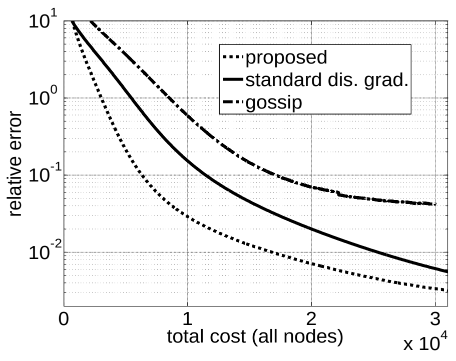
(a) Average relative error vs. number of iterations.



(b) total cost to reach rel. err. 0.01 for lower failure prob.



(c) total cost to reach rel. err. 0.01 for higher failure prob.



(d) Average relative error vs. total cost.

Fig. 4: Figures (a)-(c): Effect of asynchronous operation on the proposed method for the synthetic data set; Figure (d): comparison with the gossip-based scheme ($\alpha = \frac{1}{50\,L}$).