Random Walk Distributed Dual Averaging Method For Decentralized Consensus Optimization

Abstract

In this paper, we address the problem of distributed learning over a decentralized network, arising from scenarios including distributed sensors or geographically separated data centers. We propose a fully distributed algorithm called random walk distributed dual averaging (RW-DDA) that only requires local updates. Our RW-DDA method, improves the existing distributed dual averaging (DDA) method, making it robust to changes in network topology and amenable to asynchronous implementations. Our theoretical analysis shows the algorithm has $O(1/\sqrt{t})$ convergence for non-smooth convex problems. Various and valuable practical acceleration tricks are also introduced in the implementation. Experimental results show that our algorithm outperforms competing methods, especially in the presence of communication link failures.

1. Introduction

With technological advancements in sensors, mobile devices, and data centers, machine learning algorithms are commonly applied to data distributed across these machines. However, distributed learning in real world scenarios suffers from two issues. First, the various nodes in a distributed setting may suffer from intermittent network or node failures. For example, geographically separated data centers may suffer from communication delays or dropped packets. Second, the nodes in the distributed system such as the physical sensors may collect data points that are not randomly distributed across the nodes resulting in non-independent and identically distributed (non-i.i.d.) data across the nodes. Data-centers too, often collect non-random data, with each data center receiving data that is biased towards the geography where it is located. Often due to scale, privacy, or lack of a central coordinating resource, randomizing data may not always be possible. As a result, distributed training across these nodes with the presence of biased data at individual machines based on simple techniques such as averaging of parameters may not work.

In this paper, we propose to solve this problem in the framework of *Decentralized Consensus Optimization (DCO)*, where all the nodes (agents), with their own utility functions, are connected through a network. The networked system goal is to optimize the sum of all the utility functions, only through local computations and local information exchange with neighbors as specified by the communication graph of all nodes. Such peer-to-peer framework, with applications ranging from large scale machine learning (Tsianos et al., 2012a; Ling et al., 2012; 2013; Yuan et al., 2013b) to wireless sensor networks (Nedić & Ozdaglar, 2009; Dimakis et al., 2010), tends to be scalable, simple to implement and robust to intermittent network failures.

Given the importance of DCO, many methods have been proposed recently (Nedić, 2014). One of the techniques that gained recent popularity is a class of distributed algorithms that combine the consensus protocols developed from the control field (Olshevsky & Tsitsiklis, 2009) and the gradient-type methods from the optimization area (Nocedal & Wright, 2006). Here, a consensus protocol refers to a mechanism for information diffusion, where each agent independently spread their information via locally weighted averaging of their incoming data. The gradient-based methods are particularly suitable, since they, in general, have a small per-iteration cost and are robust to various sources of stochastic errors. In contrast with approaches based on bi-directional communication, e.g. Randomized Gossiping (Boyd et al., 2006) and ADMM-type distributed methods (Wei & Ozdaglar, 2013), the above mentioned technique employs only one-directional communication, (where agents only send information and then proceed with their local computations without expecting a response from others,) and thus manage to avoid the deadlock problem resulting from the bi-direction communication in practical implementation.

Generally speaking, based on the style of the consensus step, these methods can be classified as model averaging methods (Nedić & Ozdaglar, 2009; Ram et al., 2010; 2012; Yuan et al., 2013a; Jakovetic et al., 2014; Li et al., 2015; Shi et al., 2015), which average parameters, and dual aver-

Preliminary work. Under review by the International Conference on Machine Learning (ICML). Do not distribute.

aging methods (Duchi et al., 2012; Tsianos et al., 2012a; Tsianos & Rabbat, 2012), which average (sub)gradients. Arguably, the dual averaging approach is preferable as it is more scalable in the size of the network than the primal one (Duchi et al., 2012). However, the dual method may suffer from significant performance overheads. In specific, the dual computation and distributed consensus lead to high CPU costs when computing the dual variable every iteration, which makes it impractical when applied to large datasets as compared to primal model averaging methods.

Moreover, the successes of the above mentioned model averaging methods and dual averaging methods heavily rely on the communication network to be static, which may not be realistic due to node/edge failures. One exception is these methods (Tsianos et al., 2012b;a; Nedic & Olshevsky, 2015; Zeng & Yin, 2015) using the push-sum protocol, aka weighted gossip or sum-weight algorithms (Kempe et al., 2003; Bénézit et al., 2010; Iutzeler et al., 2013). In theory, push-sum type methods are robust to the change in network topology under the synchronous scenario. However, as pointed in (Tsianos et al., 2012a), due to the scaling issue, these methods are often numerical instable in practical asynchronous implementation. Our numerical experiment in Section 6 also conforms to this observation.

The main contribution of this paper is to propose and implement an efficient dual averaging method, which addresses the above two issues. Specifically,

- We propose an efficient algorithm, called *random walk distributed dual averaging (RW-DDA)* method, that is robust to the change in the network topology especially in presence of non-i.i.d. data where simple techniques may not work.
- We improve RW-DDA performance with an efficient implementation and discuss general stochastic subgradient optimization tricks that enable dual-space algorithms to run as fast as primal space algorithms such as model averaging.
- Finally, our experimental results demonstrate that RW-DDA can be successfully extended to asynchronous and failure-prone settings.

The paper is organized as follows: In section 2, we introduce the RW-DDA algorithm. In section 3, we analyze RW-DDA convergence. Next, in section 4 we describe extensions to RW-DDA. In sections 5 and 6, we describe our implementation and empirical evaluation. Finally, in section 7, we conclude the paper.

2. RW-DDA method for DCO

2.1. Problem Statement

In mathematical terms, the optimization problem is defined on a connected undirected network and solved by n agents (computers) collectively,

$$\min_{\boldsymbol{x}\in\mathcal{X}\subseteq\mathbb{R}^d} \quad \bar{f}(\boldsymbol{x}) := \sum_{i=1}^{n} f_i(\boldsymbol{x}). \tag{2.1}$$

The feasible set $\mathcal{X}$ is a closed and convex set in $\mathbb{R}^d$ and is commonly known by all agents, whereas $f_i : \mathcal{X} \in \mathbb{R}$ is a convex function privately known by the agent i. Throughout the paper, we also assume that f_i is L-Lipschitz continuous over $\mathcal{X}$ with respect to the Euclidean norm $\|\cdot\|$. The network $\mathcal{G} = (\mathcal{N}, \mathcal{E})$, with the node set $\mathcal{N} = [n] := \{1, 2, \cdots, n\}$ and the edge set $\mathcal{E} \subseteq \mathcal{N} \times \mathcal{N}$, specifies the topological structure on how the information can be spread among agents through local agent interactions over time. In specific, each agent i can only send and retrieve information from its neighbors $\mathcal{N}(i) := \{j \mid (j, i) \in \mathcal{E}\}$ and himself.

2.2. RW-DDA

Our random-walk distributed dual averaging (RW-DDA) method is shown step-by-step in Algorithm 1. Literally, in RW-DDA, each node i keeps a local estimate $\boldsymbol{x}_i$ and a dual variable $\boldsymbol{z}_i$ maintaining an accumulated subgradient. At iteration t, to update $\boldsymbol{z}_i$, each node needs to collect the $\boldsymbol{z}$-values of its neighbors, forms a convex combination with equal weight of the received information and adds its most recent local subgradient scaled by $|\mathcal{N}(i)| + 1$. After that, the dual variables $\boldsymbol{z}_i$ is projected to the primal space to obtain $\boldsymbol{x}_i$.

To implement RW-DDA, each node only needs to know its neighborhood information, which makes the algorithm robust to the change in network topology, frequently resulting from node failure or edge malfunction. As possibly inferred from the name, our RW-DDA method robustify the distributed dual averaging (DDA) method (Duchi et al., 2012), based on the theory of random walk over undirected graph (Ross, 1996).

Before we delve into the convergence proof for Algorithm 1, we will first make an intuitive explanation to help understand the correctness of RW-DDA.

For notational convenience, we will define the matrix $\boldsymbol{P} \in \mathbb{R}^{n\times n}$ with P_{ij} being $\frac{1}{|\mathcal{N}(i)+1|}$ for $j \in \mathcal{N}(i) \cup \{i\}$ and 0 otherwise. Clearly $\boldsymbol{P}$ is a row stochastic matrix, i.e. the sum of every row of $\boldsymbol{P}$ equals 1. We will also define the vector $\boldsymbol{\pi} \in \mathbb{R}^d$ with the i-th entry π_i being $\frac{|\mathcal{N}(i)|+1}{\beta}$, where $\beta := 2|\mathcal{V}| + |\mathcal{E}|$. It can easily verified that $\boldsymbol{\pi}$ is a probability vector, i.e. $\pi_i > 0$ and $\sum_{i\in[n]} \pi_i = 1$. With these nota-

Algorithm 1 Random Walk Distributed Dual Averaging (RW-DDA) Method

Input: a predetermined non-negative non-increasing sequence $\{\alpha(t)\}$.

Initialization: $\boldsymbol{x}_i(0) = \boldsymbol{z}_i(0) = \mathbf{0}$, for all $i \in [n]$.

for $t = 0, 1, 2, \ldots,$ **do**

1. Subgradient calculation:

$$\boldsymbol{g}_i(t) \in \partial f_i(\boldsymbol{x}_i(t)), \text{ for each agent } i. \tag{2.2}$$

2. Dual updates:

$$\boldsymbol{z}_i(t+1) = \frac{\sum_{j\in\mathcal{N}(i)\cup\{i\}} \boldsymbol{z}_j(t) + \boldsymbol{g}_i(t)}{|\mathcal{N}(i)|+1}, \tag{2.3}$$

for each agent i.

3. Primal updates:

$$\begin{aligned}\boldsymbol{x}_i(t+1) &= \mathcal{P}_{\mathcal{X}}[-\alpha(t)\boldsymbol{z}_i(t+1)] \\ &:= \arg\min_{\boldsymbol{x}\in\mathcal{X}} \|\boldsymbol{x} + \alpha(t)\boldsymbol{z}_i(t+1)\|^2,\end{aligned} \tag{2.4}$$

for each agent i.

end for

tions, we are able to express (2.3) in a terser way. Imagine $\mathcal{X} \subseteq \mathbb{R}$, so $x_i(t)$, $z_i(t)$ and $g_i(t)$ are now all scalars. Then we can rewrite the update (2.3) as

$$\begin{aligned}\boldsymbol{z}(t+1) &= \boldsymbol{P}\boldsymbol{z}(t) + \frac{1}{\beta}\mathrm{diag}\,(\boldsymbol{\pi})^{-1}\boldsymbol{g}(t) \\ &= \frac{1}{\beta}\sum_{s=0}^{t}\boldsymbol{P}^s\mathrm{diag}\,(\boldsymbol{\pi})^{-1}\boldsymbol{g}(t-s),\end{aligned} \tag{2.5}$$

with $\boldsymbol{z}(t) = (z_1(t), z_2(t), \cdots, z_n(t))^\top$ and $\boldsymbol{g}(t) = (g_1(t), g_2(t), \cdots, g_n(t))^\top$. As we need each node to play the same role in the system, from (2.5), it is quite reasonable to require $\boldsymbol{P}^\infty\mathrm{diag}\,(\boldsymbol{\pi})^{-1} = \mathbb{1}_{n\times n}$, where $\boldsymbol{P}^\infty := \lim_{t\to\infty}\boldsymbol{P}^t$ and $\mathbb{1}_{n\times n}$ is the n by n matrix with all entries as one. Indeed, we can verify this requirement by the close connection between $\boldsymbol{P}$ and $\boldsymbol{\pi}$, as revealed in the following lemma, which can be regarded as a direct consequence of results for random walk under undirected graph (Ross, 1996). This also justifies the appearance of random walk in the name of our algorithm.

Lemma 1. *$\boldsymbol{\pi}^\top\boldsymbol{P} = \boldsymbol{\pi}^\top$ and $\boldsymbol{P}^\infty := \lim_{t\to\infty}\boldsymbol{P}^t = \mathbf{1}\cdot\boldsymbol{\pi}^\top$.*

Proof. Consider a discrete-time Markov chain with state space as $\mathcal{V}$ and transition matrix specified by $\boldsymbol{P}$. It can be easily seen that this Markov chain is irreducible and aperiodic. Therefore, there exists a unique stationary distribution $\boldsymbol{d}$ satisfying $\boldsymbol{d} \geq 0$, $\mathbf{1}^\top\boldsymbol{d} = 1$, $\boldsymbol{d}^\top\boldsymbol{P} = \boldsymbol{d}^\top$ and $\boldsymbol{P}^\infty = \mathbf{1}\cdot\boldsymbol{d}^\top$. Since the probability vector $\boldsymbol{\pi}$ satisfies the so-called detailed balance equation, i.e. $\pi_i P_{ij} = \pi_j P_{ji}$, $\boldsymbol{\pi}$ is the stationary distribution, i.e. $\boldsymbol{d} = \boldsymbol{\pi}$. □

3. Convergence Analysis

In this section, we will provide an $O(1/\sqrt{t})$-convergence result for Algorithm 1 when $\alpha(t)$ is properly chosen as $O(1/\sqrt{t})$.

We first present two useful lemmas. The first one is standard in convex analysis. and the second one is from Duchi et al. (2012), which modifies slightly the result in Nesterov (2009).

Lemma 2. *For any $\boldsymbol{u}$, $\boldsymbol{v} \in \mathbb{R}^d$, $\|\mathcal{P}_{\mathcal{X}}[\boldsymbol{u}] - \mathcal{P}_{\mathcal{X}}[\boldsymbol{v}]\| \leq \|\boldsymbol{u} - \boldsymbol{v}\|$.*

Lemma 3. *Let $\{\boldsymbol{h}(t)\}_{t=1}^\infty \subset \mathbb{R}^d$ be an arbitrary sequence of vectors and $\{\alpha(t)\}_{t=1}^\infty$ be a positive and non-increasing sequence. Consider the sequences $\{\bar{\boldsymbol{z}}(t)\}_{t=0}^\infty$ with $\bar{\boldsymbol{z}}(0) = \mathbf{0}$ and $\{\boldsymbol{y}(t)\}_{t=1}^\infty$ constructed as follows:*

$$\begin{aligned}&\boldsymbol{y}(t+1) = \mathcal{P}_{\mathcal{X}}[-\alpha(t)\bar{\boldsymbol{z}}(t)], \\ &\bar{\boldsymbol{z}}(t+1) = \bar{\boldsymbol{z}}(t) + \boldsymbol{h}(t), \qquad t = 0, 1, 2, \ldots.\end{aligned}$$

Then for any $\boldsymbol{x}^\star \in \mathcal{X}$, we have

$$\begin{aligned}&\sum_{t=1}^{T}\langle\boldsymbol{h}(t), \boldsymbol{y}(t) - \boldsymbol{x}^\star\rangle \\ &\qquad\leq \frac{1}{2}\sum_{t=1}^{T}\alpha(t-1)\|\boldsymbol{h}(t)\|^2 + \frac{1}{2\alpha(T)}\|\boldsymbol{x}^\star\|.\end{aligned}$$

Now, we can proceed to our proof of the RW-DDA method.

Lemma 4. *Consider the sequences $\{\boldsymbol{x}_i(t)\}$ and $\{\boldsymbol{z}_i(t)\}$ generated by RW-DDA (Algorithm 1). Then for any $\boldsymbol{x}^\star \in \mathcal{X}$ and for each node $i \in [n]$, we have*

$$\begin{aligned}&\bar{f}\left(\widehat{\boldsymbol{x}}_i(T)\right) - \bar{f}(\boldsymbol{x}^\star) \\ &\leq \frac{L^2 n}{2\beta T}\sum_{t=1}^{T}\alpha(t-1) + \frac{\beta}{2nT\alpha(T)}\|x^\star\|^2 \\ &+ \frac{L}{T}\sum_{t=1}^{T}\alpha(t)\left(\frac{2}{n}\sum_{j=1}^{n}\|\bar{\boldsymbol{z}}(t) - \boldsymbol{z}_j(t)\| + \|\bar{\boldsymbol{z}}(t) - \boldsymbol{z}_i(t)\|\right),\end{aligned} \tag{3.1}$$

where $\widehat{\boldsymbol{x}}_i(T) = \frac{1}{T}\sum_{t=1}^{T}\boldsymbol{x}_i(t)$ and $\bar{\boldsymbol{z}}(t) = \sum_{i=1}^{n}\pi_i\boldsymbol{z}_i(t)$.

Proof. For simplicity, we assume $\mathcal{X} \subseteq \mathbb{R}$, so $x_i(t)$, $z_i(t)$ and $g_i(t)$ are now all scalars. Denote $\boldsymbol{z}(t) = (z_1(t), \ldots, z_n(t))^\top \in \mathbb{R}^d$, and $\boldsymbol{g}(t) = (g_1(t), \ldots, g_n(t))^\top \in \mathbb{R}^d$.

Based on the dynamics (2.3), we can write

$$\boldsymbol{z}(t+1) = \boldsymbol{P}\boldsymbol{z}(t) + \frac{1}{\beta}\mathrm{diag}\,(\boldsymbol{\pi})^{-1}\,\boldsymbol{g}(t). \tag{3.2}$$

Then one has the $\boldsymbol{\pi}$-weighted average

$$\begin{aligned}\bar{z}(t+1) &= \boldsymbol{\pi}^\top \boldsymbol{z}(t+1) = \boldsymbol{\pi}^\top \boldsymbol{P}\boldsymbol{z}(t) + \frac{1}{\beta}\boldsymbol{\pi}^\top \mathrm{diag}\,(\boldsymbol{\pi})^{-1}\,\boldsymbol{g}(t)\\ &= \boldsymbol{\pi}^\top \boldsymbol{z}(t) + \frac{1}{\beta}\mathbf{1}^\top \boldsymbol{g}(t) = \bar{z}(t) + \frac{1}{\beta}\mathbf{1}^\top \boldsymbol{g}(t),\end{aligned}$$

where we have used the fact that $\boldsymbol{\pi}^\top \boldsymbol{P} = \boldsymbol{\pi}^\top$ in Lemma 1.

Now define

$$y(t+1) = \mathcal{P}_{\mathcal{X}}[-\alpha(t)\bar{z}(t)], \qquad t = 0, 1, 2, \ldots, \tag{3.3}$$

and we start to bound our target $f(\widehat{x}_i(T)) - f(x^\star)$,

$$\begin{aligned}&f(\widehat{x}_i(T)) - f(x^\star)\\ &\le \frac{1}{T}\sum_{t=1}^{T}\Big(f(x_i(t)) - f(x^\star)\Big)\\ &\le \frac{1}{T}\sum_{t=1}^{T}\Big(f(y(t)) - f(x^\star)\Big) + \frac{1}{T}\sum_{t=1}^{T}\Big(f(x_i(t)) - f(y(t))\Big)\\ &\le \frac{1}{nT}\sum_{t=1}^{T}\sum_{i=1}^{n}\Big(f_i(y(t)) - f_i(x^\star)\Big) + \frac{L}{T}\sum_{t=1}^{T}\|x_i(t) - y(t)\|,\end{aligned} \tag{3.4}$$

where the first inequality is due to convexity, and last line holds as f is L-Lipschitz.

Now let us fucus on the term $f_i(y(t)) - f_i(x^\star)$,

$$\begin{aligned}&f_i(y(t)) - f_i(x^\star)\\ &= \Big(f_i(y(t)) - f_i(x_i(t))\Big) + \Big(f_i(x_i(t)) - f_i(x^\star)\Big)\\ &\le L\,\|y(t) - x_i(t)\| + \langle g_i(t), x_i(t) - x^*\rangle\\ &= L\,\|y(t) - x_i(t)\| + \langle g_i(t), x_i(t) - y(t)\rangle + \langle g_i(t), y(t) - x^\star\rangle\\ &\le L\,\|y(t) - x_i(t)\| + L\,\|y(t) - x_i(t)\| + \langle g_i(t), y(t) - x^\star\rangle\\ &\le 2L\,\|y(t) - x_i(t)\| + \langle g_i(t), y(t) - x^\star\rangle,\end{aligned} \tag{3.5}$$

where the second inequality holds as f_i is L-Lipschitz and $g_i(t) \in \partial f_i(x_i(t))$, and the second last line is due to $\|g_i\| \le L$.

Substituting (3.5) into (3.4), we obtain

$$\begin{aligned}f(\widehat{x}_i(T)) - f(x^\star) &\le \frac{1}{nT}\sum_{t=1}^{T}\sum_{i=1}^{n}\langle g_i(t), y(t) - x^\star\rangle\\ &+ \frac{L}{T}\sum_{t=1}^{T}\left(\frac{2}{n}\sum_{i=1}^{n}\|y(t) - x_i(t)\| + \|y(t) - x_i(t)\|\right).\end{aligned} \tag{3.6}$$

Next, we will look at the two terms in (3.6) respectively. For the first term,

$$\begin{aligned}&\frac{1}{nT}\sum_{t=1}^{T}\sum_{i=1}^{n}\langle g_i(t), y(t) - x^\star\rangle\\ &= \frac{\beta}{nT}\sum_{t=1}^{T}\left\langle \frac{\mathbf{1}^\top \boldsymbol{g}(t)}{\beta}, y(t) - x^\star\right\rangle\\ &\le \frac{\beta}{2nT}\sum_{t=1}^{T}\alpha(t-1)\left\|\frac{\mathbf{1}^\top \boldsymbol{g}(t)}{\beta}\right\|^2 + \frac{\beta}{2nT\alpha(T)}\|x^\star\|^2\\ &\le \frac{L^2 n}{2\beta T}\sum_{t=1}^{T}\alpha(t-1) + \frac{\beta}{2nT\alpha(T)}\|x^\star\|^2,\end{aligned} \tag{3.7}$$

where the second line results from Lemma 3 (with $\mathbf{1}^\top \boldsymbol{g}(t)/\beta$ playing the role of $h(t)$ in Lemma 3).

Regarding the second term in (3.6), note that

$$\begin{aligned}&\|y(t) - x_i(t)\| = \|\mathcal{P}_{\mathcal{X}}[-\alpha(t)\bar{z}(t)] - \mathcal{P}_{\mathcal{X}}[-\alpha(t)\bar{z}_i(t)]\|\\ &\le \|-\alpha(t)(\bar{z}(t) - z_i(t))\| \le \alpha(t)\,\|\bar{z}(t) - z_i(t)\|.\end{aligned} \tag{3.8}$$

by Lemma 2.

Finally, substituting (3.7) and (3.8) into (3.6) yields our desired (3.1). □

Lemma (4) provides a nice characterization of the deviation from the optimal value over all nodes. The first two terms in (3.1) are common optimization error terms pertaining to subgradient algorithms. The third term reflects the nature of distributed optimization, where each node has its own estimate of the average gradient that deviates from each other. Next, we will show an upper bound for the deviation term $\|\bar{\boldsymbol{z}}(t) - \boldsymbol{z}_i(t)\|$.

Lemma 5. *Consider the sequences $\{\boldsymbol{x}_i(t)\}$ and $\{\boldsymbol{z}_i(t)\}$ generated by RW-DDA (Algorithm 1). Define $\bar{\boldsymbol{z}}(t) = \boldsymbol{\pi}^\top \boldsymbol{z}(t)$. Then we have,*

$$\|\bar{\boldsymbol{z}}(t) - \boldsymbol{z}_i(t)\| \le \frac{L}{\beta\pi_{\min}}\sqrt{\frac{1-\pi_i}{\pi_i}}\frac{1}{1-\sigma_2(\boldsymbol{P})}, \tag{3.9}$$

where $\pi_{\min} = \min\{\pi_i\}$ and $\sigma_2(\cdot)$ denotes the second largest singular value.

Proof. For simplicity, we assume $\mathcal{X} \subseteq \mathbb{R}$, so $x_i(t)$ and $z_i(t)$ are now scalars. Denote $\boldsymbol{z}(t) = (z_1(t), z_2(t), \cdots, z_n(t))^\top$ and $\boldsymbol{g}(t) = (g_1(t), g_2(t), \cdots, g_n(t))^\top$. Also, in the following proof, we will omit the superscript w for notational convenience.

Due to (2.3), we have, for $t = 1, 2, \ldots,$

$$\begin{aligned}\boldsymbol{z}(t) &= \frac{1}{\beta}\mathrm{diag}\left(\boldsymbol{\pi}\right)^{-1}\boldsymbol{g}(t-1) + \boldsymbol{P}\boldsymbol{z}(t-1)\\ &= \frac{1}{\beta}\sum_{s=1}^{t}\boldsymbol{P}^{s-1}\mathrm{diag}\left(\boldsymbol{\pi}\right)^{-1}\boldsymbol{g}(t-s). \qquad (3.10)\end{aligned}$$

So,

$$\begin{aligned}z_i(t) &= \frac{1}{\beta}\sum_{s=1}^{t}\boldsymbol{e}_i^\top\boldsymbol{P}^{s-1}\mathrm{diag}\left(\boldsymbol{\pi}\right)^{-1}\boldsymbol{g}(t-s), \quad \text{and}\\ \bar{z}(t) &= \boldsymbol{\pi}^\top\boldsymbol{z}(t) = \frac{1}{\beta}\sum_{s=1}^{t}\boldsymbol{\pi}^\top\boldsymbol{P}^{s-1}\mathrm{diag}\left(\boldsymbol{\pi}\right)^{-1}\boldsymbol{g}(t-s)\\ &= \frac{1}{\beta}\sum_{s=1}^{t}\boldsymbol{\pi}^\top\mathrm{diag}\left(\boldsymbol{\pi}\right)^{-1}\boldsymbol{g}(t-s). \qquad (3.11)\end{aligned}$$

Thus,

$$\begin{aligned}&\|\bar{z}(t) - z_i(t)\|\\ &= \frac{1}{\beta}\left\|\sum_{s=1}^{t}\left(\boldsymbol{\pi}^\top - \boldsymbol{e}_i^\top\boldsymbol{P}^{s-1}\right)\mathrm{diag}\left(\boldsymbol{\pi}\right)^{-1}\boldsymbol{g}(t-s)\right\|\\ &\le \frac{L}{\beta\pi_{\min}}\sum_{s=1}^{t}\left\|\boldsymbol{\pi}^\top - \boldsymbol{e}_i^\top\boldsymbol{P}^{s-1}\right\|_1. \qquad (3.12)\end{aligned}$$

Based on the Prop. 3 of (Diaconis & Stroock, 1991),

$$\left\|\boldsymbol{\pi}^\top - \boldsymbol{e}_i^\top\boldsymbol{P}^{s-1}\right\|_1 \le \sqrt{\frac{1-\pi_i}{\pi_i}}\sigma_2^{s-1}. \qquad (3.13)$$

Substitute (3.13) into (3.12), we have

$$\begin{aligned}\|\bar{z}(t) - z_i(t)\| &\le \frac{L}{\beta\pi_{\min}}\sum_{s=1}^{t}\sqrt{\frac{1-\pi_i}{\pi_i}}\sigma_2^{s-1}\\ &\le \frac{L}{\beta\pi_{\min}}\sqrt{\frac{1-\pi_i}{\pi_i}}\frac{1}{1-\sigma_2(\boldsymbol{P})},\end{aligned}$$

which completes the proof. □

Finally, we are ready to present the convergence theorem by combining Lemma 4 and Lemma 5.

Theorem 1. *Consider the sequences $\{\boldsymbol{x}_i(t)\}$ and $\{\boldsymbol{z}_i(t)\}$ generated by RW-DDA (Algorithm 1). Define the running average at each node i as $\widehat{\boldsymbol{x}}_i(T) = \frac{1}{T}\sum_{t=1}^{T}\boldsymbol{x}_i(t)$. Then for any $\boldsymbol{x}^\star \in \mathcal{X}$ with $\|\boldsymbol{x}^\star\| \le R$, and for each node $i \in [n]$, one has*

$$\bar{f}\left(\widehat{\boldsymbol{x}}_i\left(T\right)\right) - \bar{f}(\boldsymbol{x}^\star) \le \frac{2LR}{\sqrt{nT(1-\sigma_2(\boldsymbol{P})\pi_{\min}^{3/4}}} \qquad (3.14)$$

when the step size $\alpha(t)$ is chosen as $\frac{\beta(\pi_{\min})^{3/4}\sqrt{1-\sigma_2(\boldsymbol{P})}R}{4L\sqrt{n}}\cdot\frac{1}{\sqrt{t}}$.

Proof. Let us choose $\alpha(t)$ in the form of $c/\sqrt{t}$, where c is to be optimized later.

Plugging (3.9) into (3.1), we reach

$$\begin{aligned}&\bar{f}\left(\widehat{\boldsymbol{x}}_i\left(T\right)\right) - \bar{f}(\boldsymbol{x}^\star)\\ &\le \frac{L^2 n}{\beta\sqrt{T}}\cdot c + \frac{\beta}{2n\sqrt{T}}R^2\cdot\frac{1}{c} + \frac{6L^2}{\sqrt{T}\beta\pi_{\min}^{3/2}(1-\sigma_2(\boldsymbol{P}))}\cdot c\\ &\le \frac{7L^2}{\sqrt{T}\beta\pi_{\min}^{3/2}(1-\sigma_2(\boldsymbol{P}))}\cdot c + \frac{\beta}{2n\sqrt{T}}R^2\cdot\frac{1}{c}, \qquad (3.15)\end{aligned}$$

where we have used the fact that $\sum_{t=1}^{T}t^{-1/2} \le \int_{t=0}^{T}t^{-1/2}dt = \sqrt{T}$.

The claimed result holds directly as we optimize the upper bound (3.15) with respect to the parameter c. □

4. Extension

Our RW-DDA method can be easily adapted to incorporate stochastic gradients to solve an optimization problem with a convex regularizer (e.g. ℓ_1, nuclear norm). In specific, Algorithm 2, a natural modification of RW-DDA (Algorithm 1), is capable of solving

$$\min_{\boldsymbol{x}} \quad \frac{1}{n}\sum_{i=1}^{n}f_i(\boldsymbol{x}) + \phi(\boldsymbol{x}), \qquad (4.1)$$

where $\phi(\boldsymbol{x})$ is a convex regularizer. Its convergence proof follows directly from combining our analysis above and arguments used in previous literature (Xiao, 2009; Duchi et al., 2012), which we omit here.

5. Implementation

In this section, we describe RW-DDA implementation. We implement our algorithms over the MALT framework (Li et al., 2015). MALT provides distributed machine learning over shared memory for SVM-SGD and Torch. We implement RW-DDA, simple model averaging and PS-DDA over SVM SGD. We implement model averaging such that each machine calculates the partial gradient and sends it to other machines via a *push operation*. Each machine averages the received gradients in a *reduce* step and updates its model weight vector(w) locally. In our implementation, RW-DDA and model averaging communicate variables in a one-sided fashion without requiring an acknowledgment and we use the MALT's one-sided primitives over RDMA to perform this low latency communication.

For our RW-DDA implementation, on every node k, we loop over the local training examples. For every iteration t, we choose an example i, and calculate the local gradient $\boldsymbol{g}_i$ and update the current model $\boldsymbol{x}_k^t$. In distributed optimization across multiple nodes, we perform a push oper-

Algorithm 2 Generalized Random Walk Distributed Dual Averaging (GRW-DDA) Method

Input: a predetermined nonnegative nonincreasing sequence $\{\alpha(t)\}$.
Initialization: $\boldsymbol{x}_i(0) = \boldsymbol{z}_i(0) = \mathbf{0}$, for all $i \in [n]$.
for $t = 0, 1, 2, \ldots,$ **do**

1. Stochastic subgradient calculation:

$$\mathbb{E}\left[\boldsymbol{g}_i(t)\right] \in \partial f_i(\boldsymbol{x}_i(t)), \quad \text{for each agent } i. \tag{4.2}$$

2. Dual updates:

$$\boldsymbol{z}_i(t+1) = \frac{\sum_{j \in \mathcal{N}(i) \cup \{i\}} \boldsymbol{z}_j(t) + \boldsymbol{g}_i(t)}{|\mathcal{N}(i)| + 1}, \tag{4.3}$$
$$\text{for each agent } i.$$

3. Primal updates:

$$\boldsymbol{x}_i(t+1) = \text{Prox}_{t\alpha(t)\phi(\cdot)}[-\alpha(t)\boldsymbol{z}_i(t+1)] \tag{4.4}$$
$$= \arg\min_{\boldsymbol{x}} \ \frac{1}{2}\left\|\boldsymbol{x} + \alpha(t)\boldsymbol{z}_i(t+1)\right\|^2 + t\alpha(t)\phi(\boldsymbol{x}),$$
$$\text{for each agent } i.$$

end for

ation of the computed gradients and perform a reduce operation on the received gradients. In the reduce step, we sum any incoming gradient contributions (dual vectors) as $\boldsymbol{z}_k^{t'} = \sum_{j \in \mathcal{I}_k} \boldsymbol{z}_j$ and incorporate gradient $\boldsymbol{g}_i$ into dual $\boldsymbol{z}_k$ as $\boldsymbol{z}_k^{t+1} = \frac{\boldsymbol{z}_k^{t'} + \boldsymbol{g}_i}{|\mathcal{I}_k|+1}$. After processing a batch of examples on every machine (about 500-5000), we push dual gradient z_k^{t+1} via out-edges $\mathcal{O}_k$. We also choose learning rate η_k^t and apply the dual gradient $\boldsymbol{z}_k^{t+1}$ as $\boldsymbol{x}_k^{t+1} = -\eta_k^t \cdot \boldsymbol{z}_i$. Finally, each node also maintains and updates the running average or the consensus model as $\hat{\mathbf{x}}_k^{t+1} = \sum_{i=1}^{t+1} \boldsymbol{x}_k^i/(t+1) = \frac{t}{t+1}\hat{\mathbf{x}}_k^t + \frac{1}{t+1}\boldsymbol{x}_k^{t+1}$.

To improve performance, we perform the following three optimizations. First, instead of calculating the full gradient on every iteration, we only compute the sparse gradient and separately correct the regularizer (Bottou, 2012). Second, instead of sending z (or w for model averaging) after every update step to all other nodes, we send it infrequently to reduce communication costs. Each node locally processes examples (usually 500-5000), and then communicates z. We adjust the learning rate parameter η to account for the batched communication. Finally, we maintain a running sum average over the dual, and only compute this sum only during reduce (incoming z parameters). Furthermore, in our asynchronous implementation if there are no incoming dual variables (z), we skip updating the average. We describe our distributed implementation in Algorithm **??**. We find that the above optimizations give us significant speedups (over $200X$) allowing the dual space algorithms that we implement, to operate as fast as primal space algorithms.

6. Experiments

We now evaluate the RW-DDA algorithm for training SVM using the RCV1 dataset. We evaluate RW-DDA according to the following criteria:

1. *Performance:* How does RW-DDA compare with existing primal and dual methods? We evaluate performance for the case when data is not randomly distributed across the machines (non-i.i.d case) with dense and sparse networks.

2. *Fault tolerance:* How does RW-DDA behave with non-i.i.d. data and in the presence of link failures?

We perform all experiments on a four machine research cluster connected via an infiniBand backplane. We run multiple processes, across these four machines, and we refer to each process as a rank (from the HPC terminology). We run multiple ranks on each machine, especially for models with less than 1M parameters, where a single model replica is unable to saturate the network and CPU. Each machine has an Intel Xeon 8-core, 2.2 GHz IvyBridge processor with support for SSE 4.2/AVX instructions, and 64 GB DDR3 DRAM. Each machine is connected via a Mellanox Connect-V3 56 Gbps infiniBand cards. Our 56 Gbps infiniBand network architecture provides a peak throughput of slightly over 40 Gbps after accounting for the bit-encoding overhead for reliable transmission. All machines share storage using a 10 TB NFS partition that we use for loading input data. Each process loads a portion of data depending on the number of processes. For all our experiments, we partition the input data and assign positive or negative subsets to each node. Hence, we perform all our training with a sampling bias over non-i.i.d data unless mentioned otherwise. All reported times do not account the initial one-time cost for the loading the data-sets in memory. All times are reported in seconds.

We compare RW-DDA and model averaging over the MALT framework (Li et al., 2015) with the applicable optimizations described in the previous section. Model averaging is computationally efficient because of its simple update step. We also implement failure resiliency in both the algorithms by appropriately detecting the number of nodes sending parameters and correctly computing a scaling factor. We run all our experiments over six ranks. Each rank represents a process that may span multiple machines and

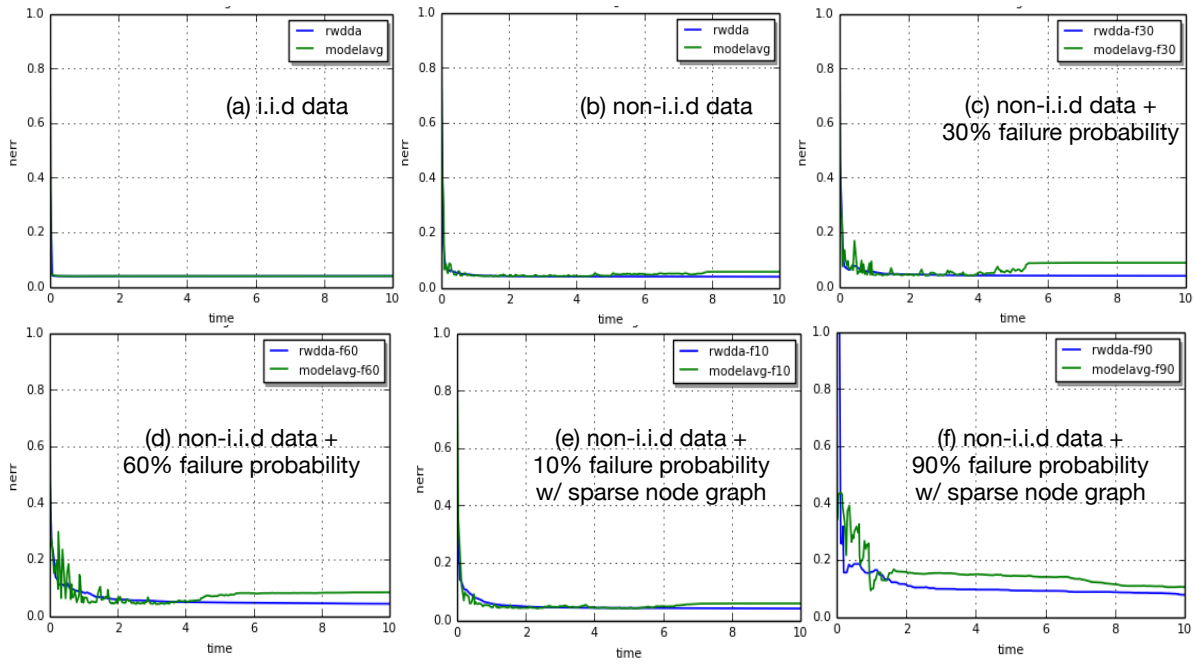

Figure 1. **This figure shows the convergence of RW-DDA with model averaging for 6 parallel ranks (processes across machines) and all ranks exchange parameters with one another. Each rank communicates z or w after processing a local epoch (slightly over 3300 examples). Figure (a) shows performance over i.i.d data where RW-DDA and model averaging compare favorably. Figure (b) shows this performance for non-i.i.d data. Figure (c) and (d) show convergence comparisons for 30% and 60% probability of packet losses. We find that RW-DDA converges in both cases. Figures (e) and (f) illustrate performance comparisons for sparse node graph where each machine only exchanges parameters with $N/2$ machines.**

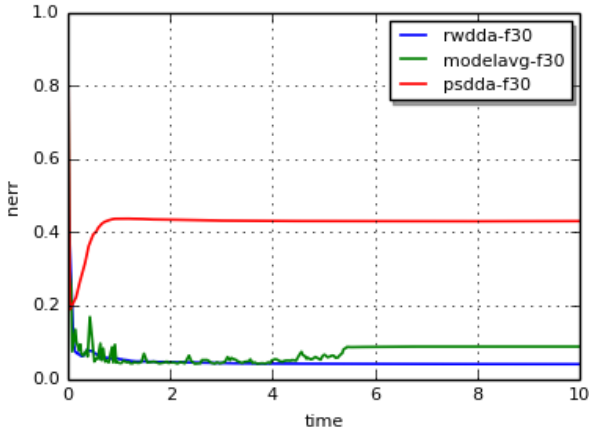

Figure 2. **This figure compares model-averaging, RW-DDA and PS-DDA for 30% probability of packet loss. We demonstrate results for average convergence across ranks and we find that PS-DDA suffers from numerical instability in the scalar that results in incorrect convergence.**

each rank trains over a subset of data. For our experiments, the six ranks span across three machines.

6.1. Performance

We compare the average training error over all ranks w.r.t. wall clock time in Figure 1 (a) and (b). In this section, we compare the performance of RW-DDA and model averaging without failures. We choose a densely connected network graph where each machine synchronizes parameters with all other machines. Figure (a) shows the convergence for i.i.d. data where both model averaging and RW-DDA converge correctly. For non-i.i.d. data, we find that RW-DDA converges faster in time, and achieves a stable accuracy better than model averaging. Hence, we find that with our optimizations, our dual-order method, RW-DDA, performs as good as primal order model averaging. From the optimizations described in Section 5, we are able to obtain more than $200X$ speedup from the original RW-DDA implementation. Furthermore, for the non-i.i.d. dataset, RW-DDA converges correctly unlike model averaging.

6.2. Fault Tolerance

We now compare RW-DDA performance in the presence of intermittent link failures. Each outgoing packet may fail with a specific user-defined probability. The failures are asymmetric i.e. nodes with positive examples are less likely to fail than those with negative examples. We re-

peat our experiments for different overall failure probability goals. For our fault tolerance experiments, we remove the barrier before the update step, since some packets may never arrive due to failures and a barrier will lead to infinite wait. Hence, we perform our fault tolerance experiments by running the algorithms asynchronously.

Figure 1 (c) and (d) show RW-DDA and model convergence with 30% and 60% packet loss probability where all machines communicate with one-another forming a dense communication graph of nodes. We find that RW-DDA is more robust to link failures and offers correct convergence which model averaging is unable to achieve. To account for fewer incoming z parameters due to the asynchrony, we appropriately re-scale the gradient or the averaging fraction by counting the number of incoming z or g parameters.

We now provide performance comparisons with undirected sparse communication graphs i.e. where all nodes may not communicate with one another. Instead of communicating with all other machines (or processes), each machine only communicates with $N/2$ other machines such that the network graph of all machines is connected and the graph is undirected, where N is the total number of nodes. We compare RW-DDA and model-averaging over a sparse communication graph in Figure 1 (e) and (f) with 10% and 90% packet loss probability. We find that model averaging does not converge correctly in Figure 1 (f) while RW-DDA achieves correctly.

Comparisons with PS-DDA We implement Push-Sum DDA (Tsianos et al., 2012a) in our framework and apply the same optimizations as RW-DDA to improve its performance. Figure 2 compares model averaging, RW-DDA and PS-DDA for failures with 30% probability of packet loss for a specific rank, with non-i.i.d. data. PS-DDA requires sending an additional scaling component. Additionally, in the asynchronous case or in presence of failures, PS-DDA suffers from numerical instability (Tsianos et al., 2012a). In asynchronous mode, since different nodes operate at different speeds, the scalar may become very small due to repeated re-scaling. To prevent this from happening, in our implementation, we reset the scalar to its initial value (1.0) if it becomes too large or too small. As a result, of the numerical instability we find that PS-DDA is unable to converge in the presence of packet losses and non-i.i.d. data. However, we find that PS-DDA performs comparably with RW-DDA in absence of failures (not shown in figure).

To summarize, from our evaluation we find that RW-DDA has good convergence properties and our implementation of RW-DDA is robust and efficient.

7. Conclusions

Distributed learning over a large number of distributed sensors or geographically separated data centers which suffers from sampling biases and communication link failures across nodes. Existing dual averaging approaches are slow, and may not converge correctly in the presence of link-failures, which are not uncommon in distributed settings. We present RW-DDA, a distributed learning algorithm that is robust to failures. Our analysis shows the algorithm has $O(1/\sqrt{t})$ convergence for non-smooth convex problems. Our experiments show that RW-DDA converges as fast as primal averaging algorithms and provides smooth convergence.

References

Bénézit, F., Blonde, V., Thiran, P., Tsitsiklis, J., and Vetterli, M. Weighted gossip: Distributed averaging using non-doubly stochastic matrices. In *Information theory proceedings (isit), 2010 ieee international symposium on*, pp. 1753–1757. IEEE, 2010.

Bottou, L. Stochastic gradient descent tricks. In *Neural Networks: Tricks of the Trade*, pp. 421–436. Springer, 2012.

Boyd, S., Ghosh, A., Prabhakar, B., and Shah, D. Randomized gossip algorithms. *IEEE/ACM Transactions on Networking (TON)*, 14(SI):2508–2530, 2006.

Diaconis, P. and Stroock, D. Geometric bounds for eigenvalues of markov chains. *The Annals of Applied Probability*, pp. 36–61, 1991.

Dimakis, A. G., Kar, S., Moura, J., Rabbat, M. G., and Scaglione, A. Gossip algorithms for distributed signal processing. *Proceedings of the IEEE*, 98(11):1847–1864, 2010.

Duchi, J. C., Agarwal, A., and Wainwright, M. J. Dual averaging for distributed optimization: convergence analysis and network scaling. *Automatic control, IEEE Transactions on*, 57(3):592–606, 2012.

Iutzeler, F., Ciblat, P., and Hachem, W. Analysis of sum-weight-like algorithms for averaging in wireless sensor networks. *Signal Processing, IEEE Transactions on*, 61(11):2802–2814, 2013.

Jakovetic, D., Xavier, J., and Moura, J. M. Fast distributed gradient methods. *Automatic Control, IEEE Transactions on*, 59(5): 1131–1146, 2014.

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

Li, H., Kadav, A., Kruus, E., and Ungureanu, C. Malt: distributed data-parallelism for existing ml applications. In *Proceedings of the Tenth European Conference on Computer Systems*. ACM, 2015.

Ling, Q., Xu, Y., Yin, W., and Wen, Z. Decentralized low-rank matrix completion. In *Acoustics, Speech and Signal Processing (ICASSP), 2012 IEEE International Conference on*, pp. 2925–2928. IEEE, 2012.

Ling, Q., Wen, Z., and Yin, W. Decentralized jointly sparse optimization by reweighted minimization. *Signal Processing, IEEE Transactions on*, 61(5):1165–1170, 2013.

Nedić, A. Distributed optimization. In *Encyclopedia of Systems and Control*, pp. 1–12. Springer London, 2014.

Nedić, A. and Ozdaglar, A. Distributed subgradient methods for multi-agent optimization. *Automatic Control, IEEE Transactions on*, 54(1):48–61, 2009.

Nedic, Angelia and Olshevsky, Alex. Distributed optimization over time-varying directed graphs. *Automatic Control, IEEE Transactions on*, 60(3):601–615, 2015.

Nesterov, Y. Primal-dual subgradient methods for convex problems. *Mathematical programming*, 120(1):221–259, 2009.

Nocedal, J. and Wright, S. *Numerical optimization*. Springer Science & Business Media, 2006.

Olshevsky, A. and Tsitsiklis, J. N. Convergence speed in distributed consensus and averaging. *SIAM Journal on Control and Optimization*, 48(1):33–55, 2009.

Ram, S. S., Nedić, A., and Veeravalli, V. V. Distributed stochastic subgradient projection algorithms for convex optimization. *Journal of optimization theory and applications*, 147(3):516–545, 2010.

Ram, S. S., Nedić, A., and Venugopal, V. V. A new class of distributed optimization algorithms: Application to regression of distributed data. *Optimization Methods and Software*, 27(1): 71–88, 2012.

Ross, S. *Stochastic processes*, volume 2. John Wiley & Sons New York, 1996.

Shi, W, Ling, Q., Wu, G., and Yin, W. Extra: An exact first-order algorithm for decentralized consensus optimization. *SIAM Journal on Optimization*, 25(2):944–966, 2015.

Tsianos, K. and Rabbat, M. G. Distributed dual averaging for convex optimization under communication delays. In *American Control Conference (ACC), 2012*, pp. 1067–1072. IEEE, 2012.

Tsianos, K., Lawlor, S., and Rabbat, M. G. Consensus-based distributed optimization: Practical issues and applications in large-scale machine learning. In *Communication, Control, and Computing (Allerton), 2012 50th Annual Allerton Conference on*, pp. 1543–1550. IEEE, 2012a.

Tsianos, K., Lawlor, S., and Rabbat, M. G. Push-sum distributed dual averaging for convex optimization. In *Decision and Control (CDC), 2012 IEEE 51st Annual Conference on*, pp. 5453–5458. IEEE, 2012b.

Wei, E. and Ozdaglar, A. On the $O(1/k)$ convergence of asynchronous distributed alternating direction method of multipliers. In *Global Conference on Signal and Information Processing (GlobalSIP), 2013 IEEE*, pp. 551–554. IEEE, 2013.

Xiao, L. Dual averaging method for regularized stochastic learning and online optimization. In *Advances in Neural Information Processing Systems*, pp. 2116–2124, 2009.

Yuan, K., Ling, Q., and Yin, W. On the convergence of decentralized gradient descent. *arXiv preprint arXiv:1310.7063*, 2013a.

Yuan, K., Ling, Q., Yin, W., and Ribeiro, A. A linearized bregman algorithm for decentralized basis pursuit. In *Signal Processing Conference (EUSIPCO), 2013 Proceedings of the 21st European*, pp. 1–5. IEEE, 2013b.

Zeng, J. and Yin, W. Extrapush for convex smooth decentralized optimization over directed networks. *arXiv preprint arXiv:1511.02942*, 2015.