On the Convergence Rate of Average Consensus and Distributed Optimization over Unreliable Networks

Lili Su
EECS, MIT
Email: lilisu@mit.edu

***Abstract*—We consider the problems of reaching average consensus and solving consensus-based optimization over unreliable communication networks wherein packets may be dropped accidentally during transmission. Existing work either assumes that the link failures affect the communication on both directions or that the message senders *know exactly* their outgoing degrees in each iteration. In this paper, we consider directed links, and we *do not* require each node know its current outgoing degree.**

We characterize the convergence rate of reaching average consensus in the presence of packet-dropping link failures. Then we apply our robust consensus update to the classical distributed dual averaging method as the information aggregation primitive. We show that the local iterates converge to a common optimum of the global objective at rate $O(\frac{1}{\sqrt{t}})$, where t is the number of iterations, matching the failure-free performance of the distributed dual averaging method.

I. Introduction

Reaching consensus and solving distributed optimization are two closely related global tasks of multi-agent networks. In the former, every agent has a private input, and the goal of the networked agents is to reach an agreement on a value that is a function of these private inputs such as maximum, minimum, average, etc; in the latter, typically, every agent has a private cost function, and the goal is to collaboratively minimize a global objective which is a proper aggregation of these private cost functions.

Average consensus has received intensive attention [8], [10], [21] partially due to the fact that one can use average consensus as a way to aggregate agents' private information. Different strategies to robustify reaching average consensus against unreliable networks have been proposed [17], [7], [4], [20], [6]. Specifically, undirected graphs were considered in [17], [6], where the link failures affect the communication in both directions; dynamically changing data and networks are considered in [6]. Directed graphs were first considered in [7], however, only biased average was achieved. This bias was later corrected in [4], [20] via introducing auxiliary variables at each agent; however, only asymptotic convergence was shown. To the best of our knowledge, the characterization of non-asymptotic convergence rate is still lacking.

Consensus-based multi-agent optimization is an important family of distributed optimization algorithms. In a typical consensus-based multi-agent optimization problem [5], [14], [13], [19], each agent i keeps a *private* cost function $h_i : \mathcal{X} \to \mathbb{R}$, and the networked agents, as a whole, want to reach agreement on a global decision $x^* \in \mathcal{X}$ such that the average of these private cost functions is minimized, i.e.,

$$x^* \in \operatorname{argmin}_{x\in\mathcal{X}} \frac{1}{n}\sum_{i=1}^{n} h_i(x),$$

where n is the total number of agents in the system. The applications of such distributed optimization problems include distributed machine learning and distributed resource allocation. Robustifying distributed optimization against link failures has received some attention recently [5], [13]. Duchi et al. [5] assumed that each realizable link failure pattern admits a doubly-stochastic matrix. In the case wherein each agent knows the number of reliable outgoing links [13], the requirement for the doubly stochastic matrices was removed by incorporating the push-sum mechanism. However, the implementation of push-sum in [13] implicitly assumed the adoption of acknowledgement mechanism.

In this work, we consider directed links, and we *do not* require each node know its current outgoing degree. As a result of this, if a message packet is dropped over a link, the sender is not aware of this loss. This scenario arises frequently in real systems. Although acknowledge mechanisms can be incorporated to improve reliability, this may slow down the convergence due to the need for message retransmission (requiring more time for each iteration of the algorithm). We characterize the convergence rate of reaching average consensus in the presence of packet-dropping link failures, which is, to the best of our knowledge, lacking in literature. Then we apply our robust consensus update to the classical distributed dual averaging method as the information aggregation primitive. We show that the local iterates converge to a common optimum of the global objective at rate $O(\frac{1}{\sqrt{t}})$, where t is the number of iterations, matching the failure-free performance of the distributed dual averaging method.

II. Network Model and Notation

We consider a synchronous system that consists of n networked agents. The network structure is represented as a *strongly connected* graph $G(\mathcal{V}, \mathcal{E})$, where $\mathcal{V} = \{1, \ldots, n\}$ is the collection of agents, and $\mathcal{E}$ is the collection of *directed* communication links. Let $\mathcal{I}_i = \{j \mid (j,i) \in \mathcal{E}\}$ and $\mathcal{O}_i = \{j \mid (i,j) \in \mathcal{E}\}$ be the sets of incoming neighbors and outgoing neighbors, respectively, of agent i. For ease of exposition, we assume there exist no self-loops, i.e., $i \notin \mathcal{I}_i \cup \mathcal{O}_i, \forall i \in \mathcal{V}$. For $i \in \mathcal{V}$, let $d_i^o = |\mathcal{O}_i|$. The communication links are

unreliable in that packets may be dropped during transmission unexpectedly. However, a given link is *operational* at least once during B consecutive iterations, where $B \geq 1$. Similar assumption is adopted in [13], [14].

III. Robust Average Consensus

Reaching average consensus in directed networks has been intensively studied [16], [1], [12]. In particular, in Push-Sum [12], [2], each networked agent updates two coupled iterates and the ratio of these two iterates approaches the average asymptotically. The correctness of Push-Sum relies crucially on "mass preservation" (specified later) of the system. However, when the communication links suffer packet-dropping failures, the desired "mass preservation" may not hold, since the transmitted "mass" may be dropped without even being notified by the senders. Robustification method has been introduced to recover the dropped "mass" [10], [9], where auxiliary variables are introduced to *record* the total "mass" sent and delivered, respectively, through a given communication link. Only asymptotic convergence is provably guaranteed [10], [9]. In this section, we focus on characterizing the convergence rate of robust average consensus. To do that, we need to modify the robust Push-Sum proposed in [10].

A. Robust Push-Sum

In this subsection, we briefly review the Push-Sum algorithm [12], [2] and its robust variant [10]. In standard Push-Sum [12], [2], described in Algorithm 1, each agent i runs two iterates:

- value sequence $\{z_i[t]\}_{t=0}^{\infty}$, and
- weight sequence $\{w_i[t]\}_{t=0}^{\infty}$,

where $z_i[0] = y_i \in \mathbb{R}^d$ is the private input, and $w_i[0] = 1 \in \mathbb{R}$ is the initial weight of agent i. The weight sequences $\{w_i[t]\}_{t=0}^{\infty}$ are introduced to relax the need for doubly stochastic matrices. Intuitively speaking, the weights are used to correct the "bias" caused by the network structure. In each iteration of Algorithm 1, each agent i divides both the local value z_i and local weight w_i by d_i^o+1, recalling that d_i^o is the out-degree of agent i in the fixed $G(\mathcal{V}, \mathcal{E})$. Among the d_i^o+1 parts of the values fractions $\frac{z_i}{d_i^o+1}$ and the weight fractions $\frac{w_i}{d_i^o+1}$, agent i sends d_i^o parts to its outgoing neighbors and one part to itself. Upon receiving the value fractions and the weight fractions from its incoming neighbors, agent i sums them up respectively. When the communication network is reliable, it has been shown that, at each agent, the ratio of the value and the weight converges to the average of the private inputs, i.e.,

$$\lim_{t\to\infty} \frac{z_i[t]}{w_i[t]} = \frac{1}{n}\sum_{j=1}^{n} y_j, \quad \forall\ i = 1, \cdots, n. \tag{1}$$

The correctness of Push-Sum algorithm relies crucially on the *mass preservation* of the system [3], [12], which says that the total weights kept by the agents in the system sum up to n at every iteration, i.e.,

$$\sum_{i=1}^{n} w_i[t] \ = \ n, \quad \forall\ t. \tag{2}$$

Unfortunately, (2) does not hold in the presence of packet-dropping link failures. Nevertheless, as illustrated in [10] (also described below in Algorithm 2), if we are able to keep track of the dropped "mass", we are able to show that the total mass is preserved in some *augmented graph*. And, running Algorithm 2 can be viewed as running standard push-sum on this augmented graph.

Algorithm 1: Push-Sum [12], [2]

Initialization: $z_i[0] = y_i \in \mathbb{R}^d$, $w_i[0] = 1 \in \mathbb{R}$.
for $t \geq 1$ **do**
  Broadcast $\frac{z_i[t-1]}{d_i^o+1}$ and $\frac{w_i[t-1]}{d_i^o+1}$ to all outgoing neighbors;
  $z_i[t] \leftarrow \sum_{j\in\mathcal{I}_i\cup\{i\}} \frac{z_j[t-1]}{d_j^o+1}$, and $w_i[t] \leftarrow \sum_{j\in\mathcal{I}_i\cup\{i\}} \frac{w_j[t-1]}{d_j^o+1}$.
end

Algorithm 2: Robust Push-Sum [10]

Initialization: $z_i[0] = y_i \in \mathbb{R}^d$, $w_i[0] = 1 \in \mathbb{R}$, $\sigma_i[0] = \mathbf{0} \in \mathbb{R}^d$, $\tilde{\sigma}_i[0] = 0 \in \mathbb{R}$, and $\rho_{ji}[0] = \mathbf{0} \in \mathbb{R}^d$, $\tilde{\rho}_{ji}[0] = 0 \in \mathbb{R}$ for each incoming link, i.e., $j \in \mathcal{I}_i$.
for $t \geq 1$ **do**
  $\sigma_i[t] \leftarrow \sigma_i[t-1] + \frac{z_i[t-1]}{d_i^o+1}$, $\tilde{\sigma}_i[t] \leftarrow \tilde{\sigma}_i[t-1] + \frac{w_i[t-1]}{d_i^o+1}$;
  Broadcast $(\sigma_i[t], \tilde{\sigma}_i[t])$ to outgoing neighbors;
  for *each incoming link* (j, i) **do**
    if *message* $(\sigma_j[t], \tilde{\sigma}_j[t])$ *is received* **then**
      $\rho_{ji}[t] \leftarrow \sigma_j[t]$, $\tilde{\rho}_{ji}[t] \leftarrow \tilde{\sigma}_j[t]$;
    else
      $\rho_{ji}[t] \leftarrow \rho_{ji}[t-1]$, $\tilde{\rho}_{ji}[t] \leftarrow \tilde{\rho}_{ji}[t-1]$;
    end
    $z_i[t] \leftarrow \sum_{j\in\mathcal{I}_i\cup\{i\}} (\rho_{ji}[t] - \rho_{ji}[t-1])$, and $w_i[t] \leftarrow \sum_{j\in\mathcal{I}_i\cup\{i\}} (\tilde{\rho}_{ji}[t] - \tilde{\rho}_{ji}[t-1])$.
  end
end

Similar to the standard Push-Sum, in Algorithm 2, each agent i wants to share with its outgoing neighbors of its value fraction $\frac{z_i}{d_i^o+1}$ and weight fraction $\frac{w_i}{d_i^o+1}$. If agent i sends these two fractions out directly, the total mass will not be preserved. In order to recover the "mass" dropped by an incoming link, in addition to $z_i[t]$ and $w_i[t]$, in Algorithm 2 each agent i uses variable $\tilde{\sigma}_i[t]$ to record the cumulative weight (up to iteration t) sent through each outgoing link; uses variable $\sigma_i[t]$ for the corresponding quantity of the value sequence. In particular,

$$\begin{aligned} \sigma_i[t] &= \sigma[t-1] + \frac{z_i[t-1]}{d_i^o+1}, \ \text{ and} \\ \tilde{\sigma}_i[t] &= \tilde{\sigma}_i[t-1] + \frac{w_i[t-1]}{d_i^o+1}, \end{aligned} \tag{3}$$

with $\sigma_i[0] = \mathbf{0} \in \mathbb{R}^d$, and $\tilde{\sigma}_i[0] = 0 \in \mathbb{R}$. In each iteration, agent i broadcasts the tuple $(\sigma_i[t], \tilde{\sigma}_i[t])$ to all of its outgoing neighbors. To record the cumulative information *delivered* via the link (i,k), the outgoing neighbor k uses a pair of variables $\rho_{ik}[t]$ and $\tilde{\rho}_{ik}[t]$, with $\rho_{ik}[0] = \mathbf{0} \in \mathbb{R}^d$ and $\tilde{\rho}_{ik}[0] = 0 \in \mathbb{R}$. If the link (i,k) is operational, i.e., the tuple $(\sigma_i[t], \tilde{\sigma}_i[t])$ is successfully delivered, then

$$\rho_{ik}[t] = \sigma_i[t], \text{ and } \tilde{\rho}_{ik}[t] = \tilde{\sigma}_i[t].$$

Otherwise, since no new message is delivered, both $\rho_{ik}[t]$ and $\tilde{\rho}_{ik}[t]$ are unchanged. In summary, if the link is operational at a given iteration, then

$$\text{total "mass" sent} \quad = \quad \text{total "mass" delivered;}$$

Otherwise,

$$\text{total "mass" sent} \quad \neq \quad \text{total "mass" delivered.}$$

In addition, if the link (i,k) is operational at iteration t, it holds that

$$\rho_{ik}[t] - \rho_{ik}[t-1] \quad = \quad \sum_{r=t'}^{t-1} \frac{z_i[r]}{d_i^o + 1}, \quad \text{and} \tag{4}$$

$$\tilde{\rho}_{ik}[t] - \tilde{\rho}_{ik}[t-1] \quad = \quad \sum_{r=t'}^{t-1} \frac{w_i[r]}{d_i^o + 1}, \tag{5}$$

where t' is the immediately preceding iteration of t such that link (i,k) is operational. As a link is reliable at least once during B consecutive iterations, it holds that $t - t' \leq B$. Under Algorithm 2, it has been shown that [10], at each agent i,

$$\frac{z_i[t]}{w_i[t]} \quad \xrightarrow{\text{a.s.}} \quad \frac{1}{n} \sum_{i=1}^{n} y_i, \quad \text{as } t \to \infty.$$

However, no convergence rate (asymptotic or non-asymptotic) is given. Informally speaking, this is because the dynamics of the system under Algorithm 2 is not stable enough. In particular, in the augmented graph constructed in [10] (formally defined later), the two iterates "kept" by the virtual agents are reset to zero periodically and unexpectedly. This "reset" causes non-trivial technical challenges – the corresponding matrix product does not converge to a rank one matrix.

B. Convergent Robust Push-Sum

In this subsection, we propose a simple variant of Algorithm 2. We refer to our algorithm as *Convergent Robust Push-Sum*, described in Algorithm 3 – simply to emphasize the fact that a finite-time convergence rate is derived. Note that this does not mean that our Algorithm 3 is superior to Algorithm 2 [10]. Our Algorithm 3 has the same set of variables as that in Algorithm 2. For ease of exposition, we use $\sigma_i^+[t]$, $\tilde{\sigma}_i^+[t]$, $z_i^+[t]$, and $w_i^+[t]$ to emphasize the fact that they are intermediate values of corresponding quantities in an iteration.

In each iteration of our Algorithm 3, the cumulative transmitted value and weight $(\sigma_i, \tilde{\sigma}_i)$, and the local value and weight (z_i, w_i) are updated twice, with the first update being identical to that in Algorithm 2. As mentioned before, with only this first update, the dynamics in the system is not stable enough, as the two iterates "kept" by the virtual agents are reset to zero periodically and unexpectedly. This "reset" is prevented by the second update in our Algorithm 3. Intuitively speaking, in the second update, each agent pushes nonzero "mass" to the virtual agents on its outgoing links. As a result of this, the two iterates "kept" by a virtual agent will never be zero at the end of an iteration.

Algorithm 3: Convergent Robust Push-Sum

Initialization: $z_i[0] = y_i \in \mathbb{R}^d$, $w_i[0] = 1 \in \mathbb{R}$, $\sigma_i[0] = \mathbf{0} \in \mathbb{R}^d$, $\tilde{\sigma}_i[0] = 0 \in \mathbb{R}$, and $\rho_{ji}[0] = \mathbf{0} \in \mathbb{R}^d$, $\tilde{\rho}_{ji}[0] = 0 \in \mathbb{R}$ for each incoming link, i.e., $j \in \mathcal{I}_i$.

for $t \geq 1$ **do**

  $\sigma_i^+[t] \leftarrow \sigma_i[t-1] + \frac{z_i[t-1]}{d_i^o+1}$, $\tilde{\sigma}_i^+[t] \leftarrow \tilde{\sigma}_i[t-1] + \frac{w_i[t-1]}{d_i^o+1}$;

  Broadcast $\left(\sigma_i^+[t], \tilde{\sigma}_i^+[t]\right)$ to outgoing neighbors;

  for *each incoming link* (j,i) **do**

    if *message* $\left(\sigma_j^+[t], \tilde{\sigma}_j^+[t]\right)$ *is received* **then**

      $\rho_{ji}[t] \leftarrow \sigma_j^+[t]$, $\tilde{\rho}_{ji}[t] \leftarrow \tilde{\sigma}_j^+[t]$;

    else

      $\rho_{ji}[t] \leftarrow \rho_{ji}[t-1]$, $\tilde{\rho}_{ji}[t] \leftarrow \tilde{\rho}_{ji}[t-1]$;

    end

    $z_i^+[t] \leftarrow \frac{z_i[t-1]}{d_i^o+1} + \sum_{j \in \mathcal{I}_i} (\rho_{ji}[t] - \rho_{ji}[t-1])$, $w_i^+[t] \leftarrow \frac{w_i[t-1]}{d_i^o+1} + \sum_{j \in \mathcal{I}_i} (\tilde{\rho}_{ji}[t] - \tilde{\rho}_{ji}[t-1])$.

  end

  $\sigma_i[t] \leftarrow \sigma_i^+[t] + \frac{z_i^+[t]}{d_i^o+1}$, $\tilde{\sigma}_i[t] \leftarrow \tilde{\sigma}_i^+[t] + \frac{w_i^+[t]}{d_i^o+1}$, $z_i[t] \leftarrow \frac{z_i^+[t]}{d_i^o+1}$, $w_i[t] \leftarrow \frac{w_i^+[t]}{d_i^o+1}$.

end

C. Augmented Graph

The augmented graph of a given $G(\mathcal{V}, \mathcal{E})$, denoted as $G^a(\mathcal{V}^a, \mathcal{E}^a)$, is constructed as follows [20]:

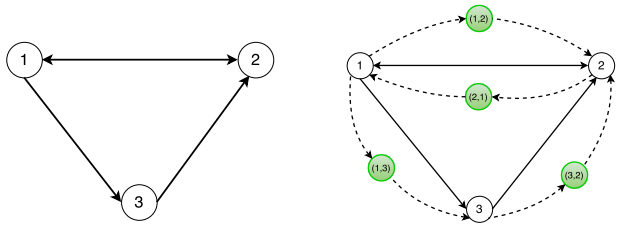
1) $\mathcal{V}^a = \mathcal{V} \cup \mathcal{E}$, i.e., $|\mathcal{E}|$ additional auxiliary agents are introduced, each of which represents a link in $G(\mathcal{V}, \mathcal{E})$. For ease of notation, we use n_{ij} to denote the virtual agent corresponding to edge (i,j).
2) $\mathcal{E} \subseteq \mathcal{E}^a$, i.e., the edge set in $G^a(\mathcal{V}^a, \mathcal{E}^a)$ preserves the topology of $G(\mathcal{V}, \mathcal{E})$;
3) Additionally, auxiliary edges are introduced: each auxiliary agent n_{ij} has one incoming neighbor – agent i – and one outgoing neighbor – agent j.

As shown in Fig. 1, in the augmented graph (i.e., Fig. 1(b)), four additional agents are introduced, each of which corresponds to a directed edge of the original graph.

D. Matrix Representation

For each link $(j,i) \in \mathcal{E}$, and $t \geq 1$, define the indicator variable $\mathsf{B}_{(j,i)}[t]$ as follows:

$$\mathsf{B}_{(j,i)}[t] \triangleq \begin{cases} 1, & \text{if link } (j,i) \text{ is reliable at time } t; \\ 0, & \text{otherwise.} \end{cases} \tag{6}$$



(a) Original graph (b) Augmented graph

Fig. 1: For each directed link, a buffer agent is added.

Recall that z_i and w_i are the value and weight for $i \in \mathcal{V} = \{1, \cdots, n\}$. For each $(j,i) \in \mathcal{E}$, we define $z_{n_{ji}}$ and $w_{n_{ji}}$ as

$$z_{n_{ji}}[t] \triangleq \sigma_j[t] - \rho_{ji}[t], \text{ and} \tag{7}$$
$$w_{n_{ji}}[t] \triangleq \tilde{\sigma}_j[t] - \tilde{\rho}_{ji}[t], \tag{8}$$

with $z_{n_{ji}}[0] = \mathbf{0} \in \mathbb{R}^d$ and $w_{n_{ji}}[0] = 0 \in \mathbb{R}$.

Let $m = n + |\mathcal{E}|$. We next show that the evolution of z and w can be described in a matrix form. Since the update of value z and weight w are identical, for ease of exposition, henceforth, we focus on the value sequence z.

From Algorithm 3, we know

$$\rho_{ji}[t] = \mathsf{B}_{(j,i)}[t]\sigma_j^+[t] + (1 - \mathsf{B}_{(j,i)}[t])\rho_{ji}[t-1], . \tag{9}$$

By (6), (7) and (9), for each $i \in \mathcal{V}$, the update of z_i is

$$\begin{cases} z_i^+[t] &= \frac{z_i[t-1]}{d_i^o+1} + \sum_{j\in\mathcal{I}_i} \mathsf{B}_{(j,i)}[t]\left(\frac{z_j[t-1]}{d_j^o+1} + z_{n_{ji}}[t-1]\right), \\ z_i[t] &= \frac{z_i^+[t]}{d_i^o+1}. \end{cases} \tag{10}$$

Thus,

$$z_i[t] = \frac{z_i[t-1]}{(d_i^o+1)^2} + \sum_{j\in\mathcal{I}_i} \frac{\mathsf{B}_{(j,i)}[t]}{(d_i^o+1)(d_j^o+1)} z_j[t-1] + \sum_{j\in\mathcal{I}_i} \frac{\mathsf{B}_{(j,i)}[t]}{(d_i^o+1)} z_{n_{ji}}[t-1]. \tag{11}$$

Similarly, we get

$$\begin{aligned} z_{n_{ji}}[t] = &\left(\frac{1}{(d_j^o+1)^2} + \frac{1-\mathsf{B}_{(j,i)}[t]}{d_j^o+1}\right) z_j[t-1] \\ &+ \sum_{k\in\mathcal{I}_j} \frac{\mathsf{B}_{(k,j)}[t]}{(d_k^o+1)(d_j^o+1)} z_k[t-1] \\ &+ \sum_{k\in\mathcal{I}_j} \frac{\mathsf{B}_{(k,j)}[t]}{d_j^o+1} z_{n_{kj}}[t-1] \\ &+ \left(1 - \mathsf{B}_{(j,i)}[t]\right) z_{n_{ji}}[t-1]. \end{aligned} \tag{12}$$

Thus, we construct a matrix $\mathbf{M}[t] \in \mathbb{R}^{m\times m}$ with the following structure:

$$\begin{aligned} &\mathbf{M}_{i,i}[t] \triangleq \frac{1}{(d_i^o+1)^2}; \\ &\mathbf{M}_{j,i}[t] \triangleq \frac{\mathsf{B}_{(j,i)}[t]}{(d_i^o+1)(d_j^o+1)}, \ \forall\ j \in \mathcal{I}_i; \\ &\mathbf{M}_{n_{ji},i}[t] \triangleq \frac{\mathsf{B}_{(j,i)}[t]}{d_i^o+1}, \ \forall\ j \in \mathcal{I}_i; \\ &\mathbf{M}_{j,n_{ji}}[t] \triangleq \frac{1}{(d_j^o+1)^2} + \frac{1-\mathsf{B}_{(j,i)}[t]}{d_j^o+1}; \\ &\mathbf{M}_{k,n_{ji}}[t] \triangleq \frac{\mathsf{B}_{(k,j)}[t]}{(d_k^o+1)(d_j^o+1)}, \quad \forall\ k \in \mathcal{I}_j; \\ &\mathbf{M}_{n_{kj},n_{ji}}[t] \triangleq \frac{\mathsf{B}_{(k,j)}[t]}{d_j^o+1}, \quad \forall\ k \in \mathcal{I}_j; \\ &\mathbf{M}_{n_{ji},n_{ji}}[t] \triangleq 1 - \mathsf{B}_{(j,i)}[t]. \end{aligned} \tag{13}$$

and any other entry in $\mathbf{M}[t]$ be zero. It is easy to check that the obtained matrix $\mathbf{M}[t]$ is row stochastic. Let $\mathbf{\Psi}(r,t)$ be the product of $t-r+1$ row-stochastic matrices

$$\mathbf{\Psi}(r,t) \triangleq \prod_{\tau=r}^{t} \mathbf{M}[\tau] = \mathbf{M}[r]\mathbf{M}[r+1]\cdots\mathbf{M}[t],$$

with $r \le t$. In addition, $\mathbf{\Psi}(t+1,t) \triangleq \mathbf{I}$ by convention.

For ease of exposition, without loss of generality, let us fix a one-to-one mapping between $\{n+1, \cdots, m\}$ and $(j,i) \in \mathcal{E}$. Thus, for each non-virtual agent $i \in \mathcal{V} = \{1, \cdots, n\}$, we have

$$z_i[t] = \sum_{j=1}^{m} z_j[0]\mathbf{\Psi}_{ji}(1,t) = \sum_{j=1}^{n} y_j \mathbf{\Psi}_{ji}(1,t), \tag{14}$$

where the last equality holds due to $z_j[0] = y_j$ for $i \in \mathcal{V}$ and $z_j[0] = 0$ for $j \notin \mathcal{V}$. Similar to (14), for the weight evolution, for each $i \in \{1, \cdots, m\}$, we have

$$w_i[t] = \sum_{j=1}^{n} w_j[0]\mathbf{\Psi}_{ji}(1,t), \tag{15}$$

Using ergodic coefficients and some celebrated results obtained by Hajnal [11], we show the following thoerem.

Theorem 1. *Under Algorithm 3, at each agent $i \in \mathcal{V} = \{1, \cdots, n\}$,*

$$\left\| \frac{z_i[t]}{w_i[t]} - \frac{1}{n}\sum_{k=1}^{n} y_k \right\| \le \frac{\sum_{k=1}^{n} y_k}{n\beta^{nB+1}} \gamma^{\lfloor \frac{t}{nB+1} \rfloor}.$$

Here we use $\|\cdot\|$ to denote ℓ_2 norm. All the missing proofs can be found in the full version [18].

IV. Robust Distributed Dual Averaging Method

We apply Algorithm 3 to distributed dual averaging method as information fusion primitive. Throughout this section, we assume that each agent i knows a private cost function $h_i : \mathcal{X} \to \mathbb{R}$, where

(A) $\mathcal{X} \subseteq \mathbb{R}^d$ is nonempty, convex and compact; and

(B) h_i is convex and L–Lipschitz continuous with respect to ℓ_2 norm, i.e., for all $x, y \in \mathcal{X}$,

$$\|h_i(x) - h_i(y)\| \leq L \|x - y\|, \forall i \in \mathcal{V} \tag{16}$$

We are interested in solving

$$\min_{x \in \mathcal{X}} \quad h(x) \triangleq \frac{1}{n} \sum_{i=1}^{n} h_i(x). \tag{17}$$

using a multi-agent network where the communication links may suffer packet-dropping failures. Let X^* be the collection of optimal solutions of h subject to $\mathcal{X}$. Since $\mathcal{X} \subseteq \mathbb{R}^d$ is a nonempty, convex and compact, X^* is also nonempty, convex and compact.

In addition to the estimate sequence $\{x[t]\}_{t=0}^{\infty}$, in dual averaging method, there is an additional sequence $\{z[t]\}_{t=0}^{\infty}$ in the dual space that essentially aggregates all the sub-gradients generated so far. In addition, the dual averaging scheme involves a *proximal function* $\psi : \mathbb{R}^d \to \mathbb{R}$ that is strongly convex. In this paper, we choose ψ to be 1–strongly convex with respect to ℓ_2 norm, that is

$$\psi(y) \geq \psi(x) + \langle \nabla \psi(x),\ y - x \rangle + \frac{1}{2} \|x - y\|^2,$$

for $x, y \in \mathbb{R}^d$. In addition, we assume that $\psi \geq 0$ and $\operatorname{argmin}_x \psi(x) = \mathbf{0} \in \mathbb{R}^d$, which is also referred as proximal center. This choice of ψ is rather standard [5], [19]. As it can be seen later, this proximal function can be used to smooth the update of the primal sequence $\{x[t]\}_{t=0}^{\infty}$.

One typical iterate sequence under dual averaging method is as follows. Initializing $z[0] = x[0] = \mathbf{0} \in \mathbb{R}^d$, for iteration ($t \geq 0$), compute $g[t] \in \partial h(x[t])$, and update z and x as

$$z[t+1] \ = \ z[t] + g[t], \tag{18}$$

$$x[t+1] \ = \ \prod\nolimits_{x \in \mathbb{R}^d}^{\psi} (z[t+1], \alpha[t]), \tag{19}$$

where $\prod_{x \in \mathbb{R}^d}^{\psi}(\cdot)$ is the projection operator defined as

$$\prod\nolimits_{x \in \mathbb{R}^d}^{\psi} (z, \alpha) \ \triangleq \ \operatorname{argmin}_{x \in \mathbb{R}^d} \left\{ \langle z, x \rangle + \frac{1}{\alpha} \psi(x) \right\}. \tag{20}$$

From (19), we know that the update of x is based on all the subgradients generated so far, and all these subgradients are weighted equally. The convergence rate of the dual averaging method is $O(\frac{1}{\sqrt{t}})$, which is faster than the subgradient method whose convergence rate is $O(\frac{\log t}{\sqrt{t}})$. Besides, the constants of the dual averaging method are often smaller [15].

Next we present our *Robust Push-Sum Distributed Dual Averaging* (RPSDA) method. In our RPSDA, each agent i locally keeps

- estimate sequence $\{x_i[t]\}_{t=0}^{\infty}$,
- gradient aggregation (value) sequence $\{z_i[t]\}_{t=0}^{\infty}$, and
- weight sequence $\{w_i[t]\}_{t=0}^{\infty}$,

where $x_i[0] = z_i[0] = \mathbf{0} \in \mathbb{R}^d$ and $w_i[0] = 1 \in \mathbb{R}$. In addition, let $\{\alpha[t]\}_{t=0}^{\infty}$ be a sequence of positive stepsizes. We will specify the choice of $\alpha[t]$ in our statement of theorem. Note that the only difference between Algorithm 4 and Algorithm 3 is that a subgradient is computed and added to the local value z. One importantly, the local estimate x is updated using dual averaging update.

Algorithm 4: RPSDA

Initialization: $z_i[0] = x_i[0] = \sigma_i[0] = \mathbf{0} \in \mathbb{R}^d$, $\tilde{\sigma}_i[0] = 0 \in \mathbb{R}$, $w_i[0] = 1 \in \mathbb{R}$, $\rho_{ji}[0] = \mathbf{0} \in \mathbb{R}^d$ and $\tilde{\rho}_{ji}[0] = 0 \in \mathbb{R}$ for each incoming link, i.e., $j \in \mathcal{I}_i$.

for $t \geq 1$ **do**

  $\sigma_i^+[t] \leftarrow \sigma_i[t-1] + \frac{z_i[t-1]}{d_i^o+1}$, $\tilde{\sigma}_i^+[t] \leftarrow \tilde{\sigma}_i[t-1] + \frac{w_i[t-1]}{d_i^o+1}$;

  Broadcast $\left(\sigma_i^+[t], \tilde{\sigma}_i^+[t]\right)$ to outgoing neighbors;

  for *each incoming link* (j, i) **do**

    if *message* $\left(\sigma_j^+[t], \tilde{\sigma}_j^+[t]\right)$ *is received* **then**

      $\rho_{ji}[t] \leftarrow \sigma_j^+[t]$, $\tilde{\rho}_{ji}[t] \leftarrow \tilde{\sigma}_j^+[t]$;

    else

      $\rho_{ji}[t] \leftarrow \rho_{ji}[t-1]$, $\tilde{\rho}_{ji}[t] \leftarrow \tilde{\rho}_{ji}[t-1]$;

    end

    $z_i^+[t] \leftarrow \frac{z_i[t-1]}{d_i^o+1} + \sum_{j \in \mathcal{I}_i} (\rho_{ji}[t] - \rho_{ji}[t-1])$, $w_i^+[t] \leftarrow \frac{w_i[t-1]}{d_i^o+1} + \sum_{j \in \mathcal{I}_i} (\tilde{\rho}_{ji}[t] - \tilde{\rho}_{ji}[t-1])$.

  end

  $\sigma_i[t] \leftarrow \sigma_i^+[t] + \frac{z_i^+[t]}{d_i^o+1}$, $\tilde{\sigma}_i[t] \leftarrow \tilde{\sigma}_i^+[t] + \frac{w_i^+[t]}{d_i^o+1}$, $z_i[t] \leftarrow \frac{z_i^+[t]}{d_i^o+1}$, $w_i[t] \leftarrow \frac{w_i^+[t]}{d_i^o+1}$.

  Compute a subgradient $g_i[t-1] \in \partial h_i(x_i[t-1])$;

  $z_i[t] \leftarrow z_i[t] + g_i[t-1]$;

  $x_i[t] \leftarrow \prod_{\mathcal{X}}^{\psi} \left(\frac{z_i[t]}{w_i[t]}, \alpha[t-1]\right)$;

end

For ease of exposition, let $g_i[r] = 0$ for each virtual agent $i \in \{n+1, \cdots, m\}$ and $r \geq 0$. Similar to (14) and (15), we have

$$z_i[t] = \sum_{r=0}^{t-1} \sum_{j=1}^{n} g_j[r] \mathbf{\Psi}_{j,i}(r, t)$$

$$w_i[t] = \sum_{j=1}^{n} \mathbf{\Psi}_{j,i}(1, t).$$

Let $\bar{z}[t] \triangleq \frac{1}{n} \sum_{i=1}^{n} z_i[t]$. We have

$$\bar{z}[t] = \frac{1}{n} \sum_{i=1}^{m} z_i[t] = \frac{1}{n} \sum_{r=0}^{t-1} \sum_{i=1}^{n} g_i[r]. \tag{21}$$

Let $\{\alpha[t]\}_{t=0}^{\infty}$ be a sequence of non-increasing stepsizes. For each agent $i \in \mathcal{V}$, we define the running average of $x_i[t]$, denoted by $\hat{x}_i[T]$, as follows:

$$\hat{x}_i[T] = \frac{1}{T} \sum_{t=1}^{T} x_i[t].$$

Theorem 2. *Let $x^* \in X^*$, and suppose that $\psi(x^*) \leq R^2$. Let $\{\alpha[t] = \frac{A}{\sqrt{t}}\}_{t=1}^{\infty}$ with $\alpha[0] = A$ be the sequence of stepsizes*

used in Algorithm 4 for some positive constant A. Then, for $T \geq nB+1$, we have for all $j \in \mathcal{V}$,

$$h\left(\hat{x}_j[T]\right) - h(x^*) \leq \frac{2L^2A}{T}(2\sqrt{T}+1) + \frac{R^2}{A\sqrt{T}} + \frac{3L^2A}{\beta^{nB+1}(1-\gamma^{\frac{1}{nB+1}})\gamma^{\frac{nB}{nB+1}}}\frac{2\sqrt{T}+1}{T}.$$

Note that Theorem 2 holds for any positive constant A. Optimizing over A, the constant hidden in $O(\frac{1}{\sqrt{T}})$ can be improved.

REFERENCES

[1] T. C. Aysal, M. E. Yildiz, A. D. Sarwate, and A. Scaglione. Broadcast gossip algorithms for consensus. *IEEE Transactions on Signal processing*, 57(7):2748–2761, 2009.

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

[3] F. Bnzit, V. Blondel, P. Thiran, J. Tsitsiklis, and M. Vetterli. Weighted gossip: Distributed averaging using non-doubly stochastic matrices. In *Proceedings of IEEE International Symposium on Information Theory Proceedings (ISIT)*, pages 1753–1757, June 2010.

[4] Y. Chen, R. Tron, A. Terzis, and R. Vidal. Corrective consensus: Converging to the exact average. In *Proceedings of IEEE Conference on Decision and Control (CDC)*, pages 1221–1228, December 2010.

[5] J. Duchi, A. Agarwal, and M. Wainwright. Dual averaging for distributed optimization: Convergence analysis and network scaling. *IEEE Transactions on Automatic Control*, 2012.

[6] I. Eyal, I. Keidar, and R. Rom. *Algorithms for Sensor Systems: 7th International Symposium on Algorithms for Sensor Systems, Wireless Ad Hoc Networks and Autonomous Mobile Entities, ALGOSENSORS 2011, Saarbrücken, Germany, September 8-9, 2011, Revised Selected Papers*, chapter LiMoSense – Live Monitoring in Dynamic Sensor Networks, pages 72–85. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012.

[7] F. Fagnani and S. Zampieri. Average consensus with packet drop communication. *SIAM Journal on Control and Optimization*, 48(1):102–133, 2009.

[8] C. N. Hadjicostis and T. Charalambous. Average consensus in the presence of delays in directed graph topologies. *IEEE Transactions on Automatic Control*, 59(3):763–768, March 2014.

[9] C. N. Hadjicostis and T. Charalambous. Average consensus in the presence of delays in directed graph topologies. *IEEE Transactions on Automatic Control*, 59(3):763–768, 2014.

[10] C. N. Hadjicostis, N. H. Vaidya, and A. D. Domínguez-García. Robust distributed average consensus via exchange of running sums. *IEEE Transactions on Automatic Control*, 61(6):1492–1507, 2016.

[11] J. Hajnal and M. Bartlett. Weak ergodicity in non-homogeneous markov chains. In *Mathematical Proceedings of the Cambridge Philosophical Society*, volume 54, pages 233–246. Cambridge Univ Press, 1958.

[12] D. Kempe, A. Dobra, and J. Gehrke. Gossip-based computation of aggregate information. In *Proceedings of IEEE Symposium on Foundations of Computer Science*, pages 482–491. IEEE, October 2003.

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

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

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

[16] R. Olfati-Saber and R. M. Murray. Consensus problems in networks of agents with switching topology and time-delays. *IEEE Transactions on Automatic Control*, 49(9):1520–1533, Sept 2004.

[17] S. Patterson, B. Bamieh, and A. El Abbadi. Distributed average consensus with stochastic communication failures. In *Proceedings of IEEE Conference on Decision and Control (CDC)*, pages 4215–4220, December 2007.

[18] L. Su and N. H. Vaidya. Robust multi-agent optimization: Coping with packet-dropping link failures. *arXiv preprint arXiv:1606.08904*, 2016.

[19] K. I. Tsianos, S. Lawlor, and M. G. Rabbat. Push-sum distributed dual averaging for convex optimization. In *Proceedings of IEEE Conference on Decision and Control (CDC)*, pages 5453–5458, December 2012.

[20] N. H. Vaidya, C. N. Hadjicostis, and A. D. Domínguez-García. Robust average consensus over packet dropping links: Analysis via coefficients of ergodicity. In *Proceedinsg of IEEE Conference on Decision and Control (CDC)*, pages 2761–2766, December 2012.

[21] L. Xiao, S. Boyd, and S.-J. Kim. Distributed average consensus with least-mean-square deviation. *Journal of Parallel and Distributed Computing*, 67(1):33–46, 2007.