Distributed Compressed Sensing For Static and Time-Varying Networks

Stacy Patterson*, Yonina C. Eldar, and Idit Keidar

***Abstract*—We consider the problem of in-network compressed sensing from distributed measurements. Every agent has a set of measurements of a signal x, and the objective is for the agents to recover x from their collective measurements using only communication with neighbors in the network. Our distributed approach to this problem is based on the centralized Iterative Hard Thresholding algorithm (IHT). We first present a distributed IHT algorithm for static networks that leverages standard tools from distributed computing to execute in-network computations with minimized bandwidth consumption. Next, we address distributed signal recovery in networks with time-varying topologies. The network dynamics necessarily introduce inaccuracies to our in-network computations. To accommodate these inaccuracies, we show how centralized IHT can be extended to include inexact computations while still providing the same recovery guarantees as the original IHT algorithm. We then leverage these new theoretical results to develop a distributed version of IHT for time-varying networks. Evaluations show that our distributed algorithms for both static and time-varying networks outperform previously proposed solutions in time and bandwidth by several orders of magnitude.**

***Index Terms*—compressed sensing, distributed algorithm, iterative hard thresholding, distributed consensus**

I. Introduction

In compressed sensing, a sparse signal $x \in \mathbb{R}^N$ is sampled and compressed into a set of M measurements, where M is typically much smaller than N. If these measurements are taken appropriately, then it is possible to recover x from this small set of measurements using a variety of polynomial-time algorithms [1].

Compressed sensing is an appealing approach for sensor networks, where measurement capabilities may be limited due to both coverage and energy constraints. Recent works have demonstrated that compressed sensing is applicable to a variety of sensor networks problems including event detection [2], urban environment monitoring [3] and traffic estimation [4]. In these applications, measurements of the signal are taken by sensors that are distributed throughout a region. The measurements are then collected at a single fusion center where signal recovery is performed. While the vast majority of recovery algorithms for compressed sensing consider a centralized setting, a centralized approach is not always feasible, especially in sensor networks where no powerful computing center is available and where bandwidth is limited.

S. Patterson is with the Department of Computer Science, Rensselaer Polytechnic Institute, Troy, NY 12180 USA. Y. Eldar, and I. Keidar are with the Department of Electrical Engineering, Technion - Israel Institute of Technology, Haifa 32000 Israel. Contact email: sep@cs.rpi.edu, phone: +972-4-829-3298, fax: +972-4-829-5757

Since the measurements are already distributed throughout a network, it is desirable to perform the signal recovery within the network itself. Distributed solutions for compressed sensing have begun to receive attention lately, and several works have proposed distributed basis pursuit algorithms for static networks where the measurement matrices are not globally known [5], [6], [7]. Although these algorithms converge to the correct solution, they do not optimize for metrics that are important in a distributed setting, most notably, bandwidth consumption. In addition, these techniques often have a high computational cost as they require every agent to solve a convex optimization problem in each iteration. Such computational capacity may not be available in low power sensor networks.

We propose an alternative approach to distributed sparse signal recovery in sensor networks that is based on *Iterative Hard Thresholding* (IHT) [8]. In a centralized setting, IHT offers the benefit of computational simplicity when compared to methods like basis pursuit. Our distributed approach maintains this same computational benefit. In addition, recent work [9] has established that centralized IHT can be used for problems beyond compressed sensing, for example sparse signal recovery from nonlinear measurements. Our distributed solution provides the same recovery guarantees as centralized IHT and thus can be applied to these additional settings as well.

In our distributed implementation of IHT, which we call DIHT, all agents store identical copies of an estimate of x. In each iteration, every agent first performs a *simple local computation* to derive an intermediate vector. The agents then perform a *global computation* on their intermediate vectors to derive the next iterate. This global computation is performed using only communication between neighbors in the network.

We present two versions of our distributed algorithm, one for static networks and one for networks with time-varying topologies. In the version for static networks, we employ standard tools from distributed computing to perform the global computation in a simple, efficient manner. The result is a distributed algorithm that outperforms previous solutions in both bandwidth and time by several orders of magnitude.

In networks that are time-varying, it is not possible to perform the global computation exactly unless each agent has a priori knowledge of the network dynamics. However, it is possible to approximate the global computation using only local communication. We first show how centralized IHT can be extended to accommodate inexact computations in each iteration while still providing the same recovery guarantees of the original IHT formulation. We then leverage these new

theoretical results to develop a version of DIHT that uses multiple rounds of a distributed consensus algorithm [10] to execute each inexact global computation in a distributed fashion. We call this algorithm *consensus-based DIHT*, or CB-DIHT. Evaluations show that CB-DIHT requires several orders of magnitude less time and bandwidth for recovery than the distributed subgradient algorithm [11], the only other algorithm for distributed compressed sensing in time-varying networks of which we are aware.

A. Related Work

Several recent works have proposed distributed compressed sensing algorithms using a basis pursuit approach. In this approach, the signal is recovered by solving a convex relaxation of the original recovery problem. These distributed methods can be divided into three classes: double-looped, single-looped, and subgradient algorithms.

The double looped techniques [12], [6] are applicable only in static networks. These algorithms consist of an inner loop, in which agents solve a dual problem, and an outer loop where the Lagrange multipliers are updated locally. In each iteration of the inner loop, the agents exchange N-vectors with their neighbors, and multiple inner loop iterations are needed to solve the dual problem.

In single-looped methods [5], in each iteration, every agent solves a local convex optimization problem; it also exchanges an N-vector with each of its neighbors and uses this vector to update the parameters of its local optimization problem. The first of these algorithms, D-Lasso [5], can be used for both distributed basis pursuit and basis pursuit denoising in a static network with synchronous messages. The second, D-ADMM [7], applies only to basis pursuit, but can be used in static networks with asynchronous messaging. We note that the convergence of D-ADMM has only been established theoretically for bipartite graphs, but experiments have shown that this algorithm converges in general graphs.

The distributed subgradient algorithm [13], [11] was proposed as a general distributed convex optimization technique but can be adapted to basis pursuit. In this approach, every agent stores an estimate of the signal. In each iteration it exchanges its estimate, an N-vector, with its neighbors and then performs a local projected subgradient step. The algorithm converges in time-varying networks, though the convergence rate has been observed to be slow.

A recent work [7] presented an experimental evaluation of the single-looped, double-looped, and subgradient methods in static networks and demonstrated that single-looped algorithms outperform the other approaches in terms of the number of vectors exchanged to converge to within a given accuracy. Of the single-looped algorithms, D-ADMM required the least amount of communication to converge. While D-ADMM uses only local communication, each agent must send its entire estimate vector to every neighbor in every iteration. This vector is not necessarily sparse until the algorithm converges, and therefore, the messages can be quite large. Furthermore, simulations show that the convergence time increases dramatically as the network connectivity increases, whereas the convergence rates of DIHT and CB-DIHT both improve with increased network connectivity.

The work by Ravazzi et al. [14] proposes a distributed compressed sensing algorithm based on iterative soft thresholding. This algorithm is similar to DIHT for static networks, however, it only converges in complete, static graphs. This is in contrast with DIHT which converges in any connected graph.

Our approach for CB-DIHT was inspired by recent work of Chen and Ozdaglar on a distributed proximal gradient technique [15]. This algorithm also simulates a centralized, inexact optimization method, in this case, a proximal gradient method, and it uses multiple iterations of distributed consensus to perform the inexact computation. This work differs from ours in that the convergence of the inexact centralized proximal gradient method had already been established [16], whereas no such prior analysis exists for centralized IHT. Furthermore, the distributed proximal gradient algorithm is only applicable to problems where the objective has a bounded gradient and where the constraints are convex. Neither of these assumptions holds in the distributed compressed sensing setting.

We base our distributed algorithm on the centralized IHT, an iterative, gradient-like algorithm for sparse signal recovery. The convergence of this centralized algorithm was established in [17], and its application to compressed sensing and related signal recovery guarantees was shown in [8]. Recently, Beck and Eldar adapted IHT to signal recovery for more general non-linear objective functions and provided theoretical guarantees on signal recovery in this setting [9]. A variation on IHT for nonlinear measurements was also proposed in [18]. This work uses a Taylor series approximation for the gradient step in each iteration rather than an exact gradient computation used in IHT in [9].

Finally, we note that, in a related work [19], we present an extension to DIHT for static networks that can further reduce bandwidth usage for problems that require many rounds to converge or when a very high recovery accuracy is desired.

B. Outline

The remainder of the paper is organized as follows. In Section II, we detail our problem setting and formulation. In Section III, we present the DIHT algorithm for static networks, and in Section IV, we present the CB-DIHT algorithm for time-varying networks. Section V gives numerical results comparing the performance of DIHT and CB-DIHT to other distributed methods for compressed sensing. Finally, we conclude in Section VI.

II. Problem Formulation

We consider a network of P agents. The agents may be the sensors themselves or they may be fusion nodes that collect measurements from several nearby sensors. We assume there is a unique agent identified as agent 1. If the uniquely identified agent is not defined a priori, one can be chosen using a variety of well-known distributed algorithms (see [20]).

The agents seek to estimate a signal $x \in \mathbb{R}^N$ that is K-sparse, meaning x has at most K non-zero elements. Each

agent has one or more (possibly noisy) measurements of the signal, and each has a loss function $f_p : \mathbb{R}^N \to \mathbb{R}$, known only to agent p, that indicates how well a given vector satisfies its measurements. The goal is for every agent to recover the signal x from their collective measurements using only communication between neighbors in the network. To recover x, the agents must solve the following optimization problem,

$$\text{minimize } f(x) := \sum_{p=1}^{P} f_p(x) \quad \text{subject to} \quad \|x\|_0 \leq K, \qquad (1)$$

where $\|\cdot\|_0$ denotes the ℓ_0 norm, i.e., the number of non-zero components. Note that each agent only has access to its own measurements, and so the agents must collaborate to solve the optimization problem.

We make the following assumption about the loss functions.

Assumption 1: The loss functions f_p, $p = 1 \dots P$, satisfy the following conditions:

(a) The function f_p is continuously differentiable over $\mathbb{R}^N$.
(b) There exists a $B_p \in \mathbb{R}$ such that for all $x \in \mathbb{R}^N$, $f_p(x) \geq B_p$.
(c) There exists a $G_p \in \mathbb{R}$ such that $\|\nabla f_p(x)\|_2 \leq G_p \|x\|$ for all $x \in \mathbb{R}^N$.
(d) The gradient ∇f_p is Lipschitz continuous over $\mathbb{R}^N$ with Lipschitz constant L_{f_p}, i.e.,

$$\|\nabla f_p(x) - \nabla f_p(y)\| \leq L_{f_p} \|x - y\|, \qquad \forall x, y \in \mathbb{R}^N.$$

The agents do not know B_p, G_p, nor L_{f_p}.

As a specific example, we consider compressed sensing [1]. Here, each agent p has M_p linear measurements of x taken using its sensing matrix $A_p \in \mathbb{R}^{M_p \times N}$. The measurement vector of agent p, denoted b_p, is given by $b_p = A_p x + e_p$, where $e_p \in \mathbb{R}^{M_p}$ is the measurement error for agent p. The loss function for each agent is

$$f_p(x) := \|A_p x - b_p\|_2^2.$$

It is straightforward to verify that this loss function satisfies Assumption 1.

We define $f(x) = \sum_{p=1}^{P} \|A_p x - b_p\|_2^2 = \|Ax - b\|_2^2$, where

$$A := \left[\begin{array}{c} A_1 \\ \hline \vdots \\ \hline A_P \end{array}\right], \qquad b := \left[\begin{array}{c} b_1 \\ \hline \vdots \\ \hline b_P \end{array}\right].$$

The objective is for agents to collaborate to solve the compressed sensing problem,

$$\text{minimize } \|Ax - b\|_2^2 \quad \text{subject to} \quad \|x\|_0 \leq K. \qquad (2)$$

In the sequel, we present solutions for the proposed *distributed sparse recovery problem* for two different network models, a static network and a time-varying network.

Static Network Model: We model the network by a directed, strongly connected graph. Agents can communicate only with their neighbors in the graph. Messaging is reliable but asynchronous, meaning that every message that is sent is eventually delivered, but the delay between sending and delivery may be arbitrarily long.

Time-Varying Network Model: Here, we consider a discrete time model. At each time step t, the network is modeled by a directed graph $(V, E^{(t)})$, where V is the set of P agents and $E^{(t)}$ are the directed communication links between them at time t. If $(q, p) \in E^{(t)}$, then agent p can send a message to agent q in time step t. Messaging is reliable and synchronous, meaning that any message sent in time t is received before time $t+1$. We adopt the following standard assumption about the network connectivity over time [21], [13], [15].

Assumption 2: The sequence of graphs $(V, E^{(t)})$, $t = 0, 1, 2, \dots$ satisfies the following conditions:

(a) The graph $(V, E^{(\infty)})$ is strongly connected, where $E^{(\infty)}$ is the set of edges that appear in infinitely many time steps.
(b) There exists an integer $C \geq 1$ such that if $(j, i) \in E^{(\infty)}$, then $(j, i) \in E^{(t)} \cup E^{(t+1)} \cup \dots \cup E^{(t+C-1)}$, for all $t \geq 0$.

In short, this assumption means that, while the network may not be connected at any given time step, the union of graphs over each interval of C time steps is a strongly connected graph. We do not require that the agents know C.

Our objective, in both network settings, is for the agents to recover the same sparse signal from their private loss functions using only local communication. In Section III, we present our distributed recovery algorithm for static networks. In Section IV, we extend this algorithm to the time-varying network model.

III. Distributed Algorithm for Static Networks

The problem (1) is known to be NP-Hard in general [22]. However, for suitable loss functions, efficient centralized algorithms exist. Our distributed recovery algorithm is based on one of these centralized algorithms, Iterative Hard Thresholding [17], [8]. We first briefly review this method and related convergence results. We then provide the details of our distributed algorithm.

A. Iterative Hard Thresholding

Consider a K-sparse signal x^* that has been measured and a loss function $f : \mathbb{R}^N \mapsto \mathbb{R}$ that captures how well a given vector matches those measurements. We assume that f is lower bounded and that it has a Lipschitz-contiuous gradient with constant L_f. IHT is a gradient-like, centralized algorithm that recovers x^* by solving the optimization problem [17], [8], [9],

$$\text{minimize } f(x) \quad \text{subject to} \quad \|x\|_0 \leq K. \qquad (3)$$

Let $\mathcal{T}_K(v)$ be the thresholding operator which returns a vector where all but the K entries of v with the largest magnitude are set to 0 (with ties broken arbitrarily). IHT begins with an initial, arbitrary K-sparse vector $x^{(0)}$ for which $f(x^{(0)})$ is finite. In each iteration, a gradient-step is performed, followed by application of the thresholding operator. This iteration is given by,

$$x^{(k+1)} = \mathcal{T}_K\left(x^{(k)} - \tfrac{1}{L}\nabla f(x^{(k)})\right), \qquad (4)$$

with $L > L_f$.

The loss function f in (3) is not necessarily convex, and, in general, optimization algorithms for non-convex objective functions only guarantee convergence to a stationary point. The inclusion of the sparsity constraint, which is also non-convex, means that we cannot employ the same definition of stationarity that is used for problems with convex constraints. We instead use a definition of a stationary point that is relevant to problem (3) called *L-stationarity* (see [9] for details).

Definition 1: For a given $L > 0$, a K-sparse vector $x^* \in \mathbb{R}^N$ is an *L-stationary point* of problem (3) if it satisfies,

$$x^* = \mathcal{T}_K \left(x^* - \tfrac{1}{L} \nabla f(x^*) \right).$$

It has been shown that L-stationarity is a necessary condition for optimality (see [9], Theorem 3.3).

With this definition, we can now state the relevant recovery result for IHT with a general nonlinear objective [9].

Theorem 3.1: Let f be lower-bounded and let ∇f be Lipschitz-continuous with constant L_f. Let $\{x^{(k)}\}_{k\geq 0}$ be the sequence generated by IHT with $L > L_f$. Then, any accumulation point of $\{x^{(k)}\}_{k\geq 0}$ is an L-stationary point of (3) .

For the compressed sensing problem, a stronger recovery result has been shown [17], [8].

Theorem 3.2: Let x^* be a K-sparse signal sampled with error e, i.e., $b = Ax^* + e$. Let $\|A\|_2 < 1$, and let A satisfy the restricted isometry property [23] with $\delta_{3K} < \frac{1}{\sqrt{32}}$. Then the sequence $\{x^{(k)}\}_{k\geq 0}$ generated by IHT with $L = 1$ satisfies

$$\|x^{(k)} - x^*\|_2 \leq 2^{-k}\|x^*\|_2 + 5\|e\|_2.$$

This theorem implies that, if the measurements are taken without error, then IHT recovers the original signal.

B. Distributed Iterative Hard Thresholding

We now present our distributed implementation of IHT for static networks. Every agent stores an identical copy of the signal estimate $x^{(k)}$. In iteration k, each agent first performs a local computation to derive an intermediate vector $z_p^{(k)} \in \mathbb{R}^N$. The agents then perform a global computation on their intermediate vectors to derive the next iterate $x^{(k+1)}$, which is, again, identical at every agent. We now define these local and global computations.

Local computation: Agent p computes its intermediate vector,

$$z_p^{(k)} = \nabla f_p(x^{(k)}), \tag{5}$$

using its local loss function and the current iterate $x^{(k)}$. This vector can be computed using only local information.

Global computation: In the global computation step, all agents must compute a function G that depends on all of their intermediate vectors. This function is defined as follows,

$$x^{(k+1)} = G\left(z_1^{(k)}, \ldots, z_P^{(k)}\right) := \mathcal{T}_K \left(x^{(k)} - \tfrac{1}{L} \textstyle\sum_{p=1}^{P} z_p^{(k)} \right). \tag{6}$$

To find G, first, the sum of the intermediate vectors is computed. In our distributed algorithm, the agents compute the sum in (6) using a well-known distributed algorithm called *broadcast/convergecast* [24] (described below). This sum is then used to complete the gradient step of the IHT iteration, followed by application of the threshold operator. The combination of the local computation step (5) and the global computation step (6) is equivalent to one iteration of the centralized IHT algorithm in (4). The details of DIHT are given below.

Algorithm 1: DIHT for agent 1.

initialize
- $k \leftarrow 0$
- $x_1^{(0)} \leftarrow x_{init}, \quad z_1^{(0)} \leftarrow 0$

while TRUE **do**
- $z_1^{(k)} \leftarrow \nabla f_1(x_1^{(k)})$
- Send $x_1^{(k)}$ to children
- Receive $sum_q^{(k)}$ from each $q \in$ children(1)
- $sum_1^{(k)} \leftarrow \left(\sum_{q \in \text{children}(1)} sum_q^{(k)} \right) + z_1^{(k)}$
- $x_1^{(k+1)} \leftarrow \mathcal{T}_K \left(x_1^{(k)} - \frac{1}{L} sum_1^{(k)} \right)$
- $k \leftarrow k + 1$

Algorithm 2: DIHT for agent $p \neq 1$.

initialize
- $k \leftarrow 0$
- $x_p^{(0)} \leftarrow 0, \quad z_p^{(0)} \leftarrow 0$

on $\texttt{receive}_p(x_1^{(k)})$ *from parent*
- $x_p^{(k)} \leftarrow x_1^{(k)}$
- $z_p^{(k)} \leftarrow \nabla f_p(x_p^{(k)})$
- Send $x_1^{(k)}$ to children
- Receive $sum_q^{(k)}$ from each $q \in$ children(p)
- $sum_p^{(k)} \leftarrow \left(\sum_{q \in \text{children}(p)} sum_q^{(k)} \right) + z_1^{(k)}$
- Send $sum_p^{(k)}$ to parent
- $k \leftarrow k + 1$

The agents first create a spanning tree over the network, rooted at agent 1, using a distributed algorithm (see [25] for details). This is done once as a pre-processing step and requires $(2|E|-(P-1))$ messages, where $|E|$ is the number of edges in the network. After the tree is constructed, each agent knows the IDs of its parent and its children in the spanning tree. Agent 1 initializes its estimate vector $x_1^{(0)}$ to x_{init}.

In each iteration k, agent 1 computes its intermediate vector $z_1^{(k)}$ according to (5). It then sends $x_1^{(k)}$ to its children. On receipt of $x_1^{(k)}$ from its parent, an agent updates its own estimate $x_p^{(k)}$ to equal $x_1^{(k)}$. It then computes $z_p^{(k)}$ by (5). The agent sends $x_1^{(k)}$ to its children, if it has any. If an agent does not have any children, it sends its vector $z_p^{(k)}$ to its parent. Once an agent has received vectors from all of its children, it adds those vectors to its own $z_p^{(k)}$ and sends the resulting sum to its parent (if it is not agent 1). When agent 1 receives vectors from all of its children, it adds these vectors to $z_1^{(k)}$ and finishes the global computation (6) to obtain $x_1^{(k+1)}$. This completes one iteration of the algorithm. The process is then

repeated to obtain the next iterate. Pseudocode for DIHT is given in Algorithms 1 and 2.

C. Algorithm Analysis

D-IHT requires $O(N)$ storage at each agent. In every iteration, each agent computes the gradient of its local loss function. In compressed sensing, this requires only matrix-vector multiplication. Therefore, the local computation is much simpler than the double and single looped algorithms that require each agent to solve a convex optimization problem in every iteration. Computing the sum of the intermediate vectors requires $3(P-1)$ total messages, and each agent p sends at most $2D_p+1$ messages, where D_p is the node degree of agent p. At most D_p+1 of these messages are N-vectors, and the remaining messages are K-sparse vectors (for the distribution of $x_1^{(k)}$). We note that, while agent 1 plays a unique role in the local and global computations, it performs the same number and types of computations has the same messaging cost as any other agent, with the exception of the thresholding operation, which can be performed in a single scan of the sum vector.

The estimates $x_p^{(k)}$, $p=1\dots P$, are equivalent to each other in iteration k, and they evolve exactly as $x^{(k)}$ in (6). Therefore, DIHT provides the same the convergence guarantees as centralized IHT. This is formalized in the following theorems.

Theorem 3.3: Let each f_p, $p=1\dots p$, satisfy Assumption 1, and let $\{x_p^{(k)}\}_{k\geq 0}$, $p=1\dots P$, be the sequences of estimates generated by DIHT with $L>\sum_{p=1}^{P} L_{f_p}$ in a static network. Then, any accumulation point of $\{x_p^{(k)}\}_{k\geq 0}$, $p=1\dots P$ is an L-stationary point of (1) .

Theorem 3.4: For the distributed compressed sensing problem (2), let x^* be the original K-sparse signal measured with error e, and let A be such that $\|A\|_2<1$ and satisfy the restricted isometry property with $\delta_{3K}<\frac{1}{\sqrt{32}}$. Then, the sequences of estimates $\{x_p^{(k)}\}_{k\geq 0}$, $p=1\dots P$, generated by DIHT with $L=1$ in a static network satisfy

$$\|x_p^{(k)}-x^*\|_2\leq 2^{-k}\|x^*\|_2+5\|e\|_2.$$

IV. Distributed Algorithm for Time-Varying Networks

We now show how to extend DIHT to networks with time-varying topologies. The local computation step (5) remains the same. For the global computation step (6), the broadcast-convergecast algorithm used to compute the sum of the intermediate vectors requires a static network; it cannot be applied in a time-varying network setting. In fact, without a priori knowledge of the network dynamics or membership, it is not possible for the agents to compute the exact sum of the intermediate vectors in finite time using *any* algorithm. This is because, without this knowledge, an agent cannot determine when it has received the information that it needs (from all other agents) to compute the sum in (6).

In our extended DIHT algorithm, we instead use a distributed consensus algorithm [10] to *approximate* the average of the intermediate vectors and then use this approximation to complete the gradient step, followed by application of the thresholding operator. Distributed consensus can be implemented without any global knowledge of the network, and it has been shown that, in networks with a time-varying topologies, distributed consensus converges to the average of the agents' initial values [21]. While the agents learn the exact average after infinite time, after a finite number of iterations, they learn an approximation of this average. To use distributed consensus for the global computation steps in DIHT, we must first consider the effects of such approximation errors on the correctness of the centralized IHT algorithm.

In Section IV-A, we present new theoretical results on the convergence of centralized IHT with approximate sums. We capture these approximations in the form of inexact computations of the gradient ∇f. We show that, under a limited assumption on the accuracy of the gradient values, IHT with inexact gradients provides the same recovery guarantees as IHT with exact gradient computations. We leverage these new theoretical results to develop a consensus-based distributed IHT algorithm for time-varying networks. In Section IV-B, we present a variation of distributed consensus that allows agent 1 to initiate the consensus algorithm in each iteration of IHT. We then present the complete CB-DIHT algorithm in Section IV-C, followed by convergence analysis in Section IV-D.

A. IHT with Inexact Gradients

The IHT algorithm with inexact gradients is initialized with a K-sparse vector $x^{(0)}$ for which $f(x^{(0)})$ is finite. In each iteration, an approximate gradient step is computed, followed by the application of the thresholding operator. The iteration is thus given by,

$$x^{(k+1)}=\mathcal{T}_K\left(x^{(k)}-\tfrac{1}{L}\left(\nabla f(x^{(k)})+\epsilon^{(k)}\right)\right), \qquad (7)$$

with $L<L_f$. Here, $\epsilon^{(k)}\in\mathbb{R}^N$ the error in the gradient computation in iteration k.

The following theorems show that, so long as the sequence $\{\|\epsilon^{(k)}\|^2\}_{t\geq 0}$ is summable, algorithm (7) provides the same convergence guarantees as IHT with exact gradients in (4). Proofs are deferred to Appendix A. Our first theorem (analogous to Theorem 3.1 in [9]) states that any accumulation point is an L-stationary point (as defined in Definition 1).

Theorem 4.1: Let f be lower-bounded and let ∇f be Lipschitz-continuous with constant L_f. Let $\{x^{(k)}\}_{k\geq 0}$ be the sequence generated by algorithm (7) with $L>L_f$ and with a sequence $\{\epsilon^{(k)}\}_{k\geq 0}$ satisfying $\sum_{k=0}^{\infty}\|\epsilon^{(k)}\|^2<\infty$. Then, any accumulation point of $\{x^{(k)}\}_{k\geq 0}$ is an L-stationary point.

For the compressed sensing problem, we can show a stronger result (analogous to Theorem 3.2 in [9]). Let $f(x)=\|Ax-b\|^2$. We define the matrix A to be *K-regular* if, for every index set $I\subseteq\{1,2,\dots,N\}$ with $|I|=K$, the columns of A associated with the index set I are linearly independent. If A is K-regular, then algorithm (7) converges to an L-stationary point.

Theorem 4.2: Let $f(x)=\|Ax-b\|^2$, with A satisfying the K-regularity property. Let $\{x^{(k)}\}_{k\geq 0}$ be the sequence generated by (7) with $L>2\lambda_{max}(A^{\mathsf{T}}A)$ and with a sequence $\{\epsilon^{(k)}\}_{t\geq 0}$ satisfying $\sum_{k=0}^{\infty}\|\epsilon^{(k)}\|^2<\infty$. Then, the sequence $\{x^{(k)}\}_{k\geq 0}$ converges to an L-stationary point.

Algorithm 3: Diffusive Distributed Consensus algorithm for agent p.

initialize
  if $p = 1$ **then**
    initiated $\leftarrow$ TRUE
  else
    initiated $\leftarrow$ FALSE
  activeNeighbors $\leftarrow \emptyset$
  $v_p^{(0)} \leftarrow$ initial value
for $t = 0 \ldots \infty$ **do**
  if *initiated* = TRUE **then**
    $\mathcal{N}_p(t) \leftarrow$ all agents $q \in$ *activeNeighbors* where $(p,q) \in E^{(t)}$
    $v_p^{(t+1)} \leftarrow \sum_{q \in \mathcal{N}_p(t)}^{P} w_{pq}^{(t)}\ v_q^{(t)}$
    for $(q,p) \in E^{(t)}$ *and* $q \notin$ *activeNeighbors* **do**
      send INITIATE to q
      activeNeighbors $\leftarrow$ *activeNeighbors* $\cup \{q\}$
  if *receive* INITIATE *from some agent* q **then**
    if *initiated* = FALSE **then**
      initiated $\leftarrow$ TRUE
    activeNeighbors $\leftarrow$ *activeNeighbors* $\cup \{q\}$

B. Distributed Diffusive Consensus

As previously stated, in each iteration of CB-DIHT, the agents use distributed consensus to compute an approximation of the average of their intermediate vectors $z_p^{(k)}$, $p = 1 \ldots P$. In the standard formulation of distributed consensus in a time-varying network, every agent has an initial, vector-valued state $v_p^{(0)}$. In time step t, every agent computes a weighted average of its value and that of its neighbors in that time step. The vector at agent p evolves as,

$$v_p^{(k+1)} = \sum_{q=1}^{P} w_{pq}^{(k)}\ v_q^{(k)}, \tag{8}$$

where $w_{pq}^{(t)}$ is the weight that agent p assigns to the value at agent q. Under appropriate assumptions about the choice of weights and the connectivity of the network over time (e.g. Assumption 2), the agents' vectors converge to the average of the initial vectors $v_{av} = \frac{1}{P}\sum_{p=1}^{P} v_p^{(0)}$ [21].

In CB-DIHT, the agents need to compute an approximate average in each iteration. As with DIHT for static networks, agent 1 initiates this global computation and computes the next iterate once the global computation is complete. To use distributed consensus in the global computation step of CB-DIHT, we must augment the standard consensus algorithm so that it can be initiated by a single agent, just as agent 1 initiated the broadcast/convergcast algorithm DIHT for static networks. We now explain the details of our modified consensus algorithm, which we call *diffusive distributed consensus*.

The algorithm operates in discrete time steps. In any time step, an agent may be *initiated*, meaning it is participating in the consensus algorithm and sends information along all its outgoing links in that time step, or it may be *uninitiated*, meaning it is not yet participating in the algorithm. We call a link (q,p) *active* at time t if agents p and q were initiated prior to time t. We assume that agent 1 begins the algorithm at time step 0 and thus is the only initiated agent at that time. In step 0, agent 1 sends an INITIATE message along its outgoing links, $i.e.$, it sends messages to all agents p such that $(p,1) \in E^{(0)}$. Upon receipt of this message, each agent is initiated. In step 1, the initiated agents begin the consensus algorithm specified by (8) over the active links that are present in that time step. When an agent p is initiated in a time step k, it initiates its uninitiated neighbors (for which a link is present) in time step $k+1$ by sending an INITIATE message to them. It also performs the consensus iteration (8) over its active links that are present in that time step. In this manner, the initiation process diffuses through the network until the entire network is participating in the consensus algorithm, at which point the algorithm is identical to standard distributed consensus.

Pseudocode for the diffusive distributed consensus algorithm is given in Algorithm 3. One can think of the diffusive distributed averaging consensus as a standard distributed consensus algorithm over a graph $(V, \overline{E}^{(t)})$, where $\overline{E}^{(t)} \subseteq E^{(t)}$ contains only active edges.

To ensure convergence of the diffusive distributed consensus algorithm, we require that the time-varying network satisfy the network connectivity conditions stated in Assumption 2. A consequence of this assumption is that for $\overline{C} = \Delta C$, where Δ is the diameter of the graph $(V, E^{(\infty)})$, for all $(q,p) \in E^{(\infty)}$, we have $(q,p) \in \overline{E}^{(k)} \cup \overline{E}^{(k+1)} \cup \cdots \cup \overline{E}^{(k+\overline{C}-1)}$, for all $k \geq 0$. We also make the following standard assumption on the weights used in iteration (8) [21], [13].

Assumption 3: The weight matrices $W(k) := [w_{pq}^{(k)}]$, $k = 0, 1, 2, \ldots$ satisfy the following conditions:

(a) $w_{pq}^{(k)}$ only if $(p,q) \in \overline{E}^{(k)}$.
(b) The matrix $W^{(k)}$ is doubly stochastic.
(c) There exists a scalar $\eta \in (0,1)$, such that, for all p, $w_{pp}^{(k)} \geq \eta$. Additionally, if $(p,q) \in \overline{E}^{(k)}$, then $w_{pq}^{(k)} \geq 0$, and if $(p,q) \notin \overline{E}^{(k)}$, then $w_{pq}^{(k)} = 0$.

Under Assumptions 2 and 3, we can bound the deviation between the average v_{av} and any agent's estimate of that average after s iterations of diffusive distributed consensus.

Theorem 4.3: Let the network satisfy Assumption 2, and let Δ be the diameter of the graph $(V, E^{(\infty)})$. After s iterations of diffusive distributed consensus, initiated at a single agent, where the weights obey Assumption 3, the deviation of each agent's estimate from the average v_{av} satisfies,

$$\|v_p^{(s)} - v_{av}\| \leq \Gamma\gamma^s \sum_{q=1}^{P} \|v_q^{(0)}\|, \quad \text{for } p = 1, \ldots P,$$

where $\Gamma = 2(1+\eta^{-\overline{D}})/(1-\eta^{\overline{D}})$ and $\gamma = (1-\eta^{\overline{D}})^{1/\overline{D}}$, with $\overline{D} = \left(|E^{(\infty)}| - 1\right)\Delta C$.
This theorem is a straightforward extension of Proposition 1 in [13] for the standard distributed consensus algorithm.

C. Consensus-Based DIHT

We now detail our CB-DIHT algorithm. As in DIHT for static networks, each agent has an estimate $x_p^{(k)}$, and the agents

Algorithm 4: Consensus-Based DIHT for agent 1.

initialize

$x_p^{(0)} \leftarrow x_{init}$

$k \leftarrow 0$

while TRUE **do**

$z_1^{(k)} \leftarrow \nabla f_1(x^{(k)})$ *Local computation.*

$s^{(k)} \leftarrow \left\lceil \frac{k+\|x^{(k)}\|^2}{2} \right\rceil$ *Number of consensus steps.*

$v_1^{(k)} \leftarrow \texttt{DiffusiveAveraging}(z_1^{(k)}, s^{(k)})$

$x_1^{(k+1)} \leftarrow \mathcal{T}_{\kappa}\left(x_k^{(1)} - \frac{1}{L} v_1^{(k)}\right)$

$k \leftarrow k+1$

Algorithm 5: Consensus-Based DIHT for agent $p \neq 1$.

initialize

$x_p^{(k)} \leftarrow 0$

$z_p^{(k)} \leftarrow \nabla f_p(x_p^{(k)})$

$k \leftarrow 0$

while TRUE **do**

on $\texttt{receive}_p(x_1^{(k)})$ *for first time*

stop averaging for iteration $k-1$

$x_p^{(k)} \leftarrow x_1^{(k)}$

$z_p^{(k)} \leftarrow \nabla f_p(x_p^{(k)})$

send $x_1^{(k)}$ to neighbors (excluding q).

$k \leftarrow k+1$

start averaging for $z_p^{(k)}$ *in next time step*

maintain identical estimates as the algorithm progresses. For each iteration k of CB-DIHT, agent 1 computes its intermediate vector, according to (5). It then initiates the diffusive distributed consensus algorithm for iteration k by sending $x_1^{(k)}$ along its outgoing links. All agents use the vector $x_1^{(k)}$ as the initiation message for this instance of the consensus algorithm. When an agent $p \neq 1$ receives an initiation message containing $x_1^{(k)}$, it ceases participating in the diffusive consensus algorithm instance for iteration $k-1$ (if applicable). It then computes $z_p^{(k)}$ according to (5), and it begins participating in the diffusive consensus algorithm instance for iteration k of CB-DIHT.

After agent 1 executes $s^{(k)} = \lceil (k + \|x_1^{(k)}\|^2)/2 \rceil$ time steps of diffusive distributed consensus, it uses its local estimate of the average, denoted $\hat{v}^{(t)}$, to compute $x_1^{(k+1)}$ as follows,

$$x_1^{(k+1)} = \mathcal{T}_{\kappa}\left(x_1^{(k)} - \tfrac{1}{L}\hat{v}^{(k)}\right). \tag{9}$$

It then begins a new instance of diffusive distributed consensus for iteration $k+1$ of CB-DIHT. The pseudocode for CB-DIHT is given in Algorithms 4 and 5.

D. Algorithm Analysis

CB-DIHT requires $O(N)$ storage at each agent. In each round of diffusive distributed consensus, every agent sends its estimate, an N-vector, along all outgoing active links. With respect to computational complexity, each agent must compute its local gradient, which, in the case of compressed sensing, consists of matrix-vector multiplication. Agent 1 performs the thresholding operation which requires a single scan of the vector $\hat{v}^{(k)}$.

For each iteration k, the estimates at all agents $p \neq 1$ are identical to those at agent 1, and so it suffices to analyze the convergence of the estimate at agent 1. This estimate evolves according to the following equations,

$$s^{(k)} = \left\lceil \frac{k+\|x_1^{(k)}\|^2}{2} \right\rceil \tag{10}$$

$$\hat{v}^{(k)} = \sum_{p=1}^{P} \left[\Phi(s^{(k)})\right]_{1p} \nabla f_p(x_1^{(k)}) \tag{11}$$

$$x_1^{(k+1)} = \mathcal{T}_{\kappa}\left(x_1^{(k)} - \tfrac{1}{L}\hat{v}^{(k)}\right), \tag{12}$$

where $\Phi(s(k))$ is the product of the weight matrices for $s^{(k)}$ time steps of the diffusive distributed consensus algorithm, i.e.,

$$\Phi(s(k)) = W^{(k_{s^{(k)}})} W^{(k_{s^{(k)}-1})} \cdots W^{(k_2)} W^{(k_1)},$$

with each $W^{(k_i)}$ satisfying Assumption 3. The notation $[\cdot]_{1p}$ indicates the entry of matrix at row 1, column p. The vector $\hat{v}^{(k)}$ is thus agent 1's estimate of the average of the intermediate vectors in iteration k of CB-DIHT.

It is straightforward to show that the evolution of $x_1^{(k)}$ can be formulated as an execution of centralized IHT with inexact gradients.

Proposition 4.4: The evolution of the estimate $x_1^{(k)}$ specified by (10) - (12) can be written as

$$x_1^{(k+1)} = \mathcal{T}_{\kappa}\left(x_1^{(k)} - \tfrac{1}{L}\left(\nabla f(x_1^{(k)}) + \epsilon^{(k)}\right)\right),$$

where $f = \sum_{p=1}^{P} \nabla f_p(x_1^{(k)})$ and

$$\epsilon^{(k)} = \sum_{p=1}^{P} \nabla f_p(x_1^{(k)}) - \hat{v}^{(k)}.$$

We now show that, under Assumptions 1, 2, and 3 and an appropriate choice of L, the sequence of approximation errors is square-summable.

Lemma 4.5: Under Assumptions 1, 2, and 3, the sequence $\{\epsilon^{(k)}\}_{k\geq 0}$ defined in Proposition 4.4 satisfies,

$$\sum_{k=0}^{\infty} \|\epsilon^{(k)}\|_2^2 < \infty. \tag{13}$$

Proof: Let $\overline{v}^{(k)}$ denote the exact average of the intermediate vectors in iteration k of CB-DIHT, i.e.,

$$\overline{v}^{(t)} = \sum_{p=1}^{P} \nabla f_p(x_1^{(t)}),$$

We can thus express $\epsilon^{(k)}$ as

$$\epsilon^{(k)} = \overline{v}^{(k)} - \hat{v}^{(k)}$$

Using Theorem 4.3, we can bound the difference between $\overline{v}^{(k)}$ and $\hat{v}^{(k)}$,

$$\|\overline{v}^{(k)} - \hat{v}^{(k)}\| \leq \Gamma \gamma^{s^{(k)}} \sum_{q=1}^{P} \nabla f_q(x_1^{(k)}).$$

Combined with (13), we have,

$$\sum_{k=0}^{\infty} \|\overline{v}^{(k)} - \hat{v}^{(k)}\|^2 \leq \sum_{k=0}^{\infty} \left(\Gamma \gamma^{s^{(k)}} \sum_{q=1}^{P} \|\nabla f_p(x^{(k)})\| \right)^2 \quad (14)$$

$$\leq \sum_{k=0}^{\infty} \left(\Gamma \gamma^{s^{(k)}} G \|x_1^{(k)}\| \right)^2 \quad (15)$$

$$\leq \Gamma^2 G^2 \sum_{k=0}^{\infty} \gamma^{2s^{(k)}} \|x_1^{(k)}\|^2, \quad (16)$$

with $G = \sum_{p=1}^{P} G_p$. Here, (15) follows from (14) by Assumption 1(c).

Substituting $s^{(k)} = \lceil (k + \|x_k^{(1)}\|^2)/2 \rceil$ into (16), we obtain

$$\sum_{k=0}^{\infty} \|\overline{v}^{(k)} - \hat{v}^{(k)}\|^2 \leq \Gamma^2 G^2 \sum_{k=0}^{\infty} \gamma^{2\lceil (k+\|x^{(k)}\|^2)/2 \rceil} \|x^{(k)}\|^2$$

$$\leq \Gamma^2 G^2 \sum_{k=0}^{\infty} \gamma^k \|x^{(k)}\|^2 \gamma^{\|x^{(k)}\|^2}.$$

We note that since $\gamma \in (0,1)$, the function $x\gamma^x$ is bounded for all $x \in \mathbb{R}$. Thus, the sequence $\{\|\overline{v}^{(k)} - \hat{v}^{(k)}\|^2\}_{k\geq 0}$ is upper-bounded by the geometric sequence $\{H\gamma^k\}_{t\geq 0}$ for some constant H. Since $\gamma \in (0,1)$, this geometric sequence is summable, and thus the sequence $\{\|\overline{v}^{(k)} - \hat{v}^{(k)}\|^2\}_{k\geq 0}$ is also summable, proving the theorem. ∎

The following theorem follows directly from Proposition 4.4, Lemma 4.5, and Theorems 4.1 and 4.2.

Theorem 4.6: Let Assumptions 1, 2, and 3 hold, and let $\{x_p^{(k)}\}_{k\geq 0}$, $p = 1 \dots P$, be the sequences generated by CB-IHT with $L > \frac{1}{P}\sum_{p=1}^{P} L_{f_p}$. Then, the following hold:

1) Any accumulation point of the sequence $\{x_p^{(k)}\}_{k\geq 0}$, $p = 1 \dots P$, is an L-Stationary point of (1).
2) For the compressed sensing problem (2), where A satisfies the K-regularity property, the sequences $\{x_p^{(k)}\}_{k\geq 0}$, $p = 1 \dots P$, converge to an L-stationary point.

Furthermore, if a sequence $\{x_p^{(k)}\}_{k\geq 0}$, $p = 1 \dots P$, converges to an L-stationary point x^*, then all other sequences, $\{x_q^{(k)}\}_{k\geq 0}$, $q = 1 \dots P, q \neq p$, also converge to x^*.

V. Simulations

In this section, we present an experimental comparison of DIHT and CB-DIHT with previously proposed distributed algorithms for sparse signal recovery. Note that, while DIHT and CB-DIHT can be used to recover signals from nonlinear measurements, we are unaware of any other distributed methods that address this general problem. Therefore we evaluate how each algorithm performs on the distributed compressed sensing example (2), for which there are several other existing algorithms. We now briefly review the other algorithms that we use for comparison.

A. Other Algorithms

As discussed in Section I-A, all of the other algorithms for distributed compressed sensing use an approach based on the convex relaxation of problem (2), for example a basis pursuit formulation [26],

$$\begin{aligned} &\text{minimize } \tfrac{1}{P}\textstyle\sum_{p=1}^{P} \|x\|_1 \\ &\text{subject to } A_p x = b_p, \quad p = 1 \dots P. \end{aligned} \quad (17)$$

In a recent work, Mota et al. compared several distributed basis pursuit algorithms for static networks, and they showed that D-ADMM outperformed all other approaches in terms of the number of messages [7]. We therefore use D-ADMM as the representative example of convex relaxation-based methods in our evaluation.

For time-varying networks, we are aware of only one other distributed algorithm for problem (2), the distributed subgradient algorithm [11]. This algorithm was proposed to solve a general class of convex optimization problems, of which, the convex relaxation (17) is a special case.

We now briefly describe each of these algorithms. Pseudocode is given in Appendix B.

1) D-ADMM: As it's name suggests, D-ADMM is a distributed version of the alternating direction method of multipliers. The algorithm requires that a graph coloring is available, meaning that every agent is assigned a color such that no two neighboring agents share the same color. Each agent has its own estimate, $x_p^{(k)}$. It waits to receive estimates from all of its neighbors with lower colors, and it computes an intermediate vector based on the last received estimates from all neighboring agents. It then uses this intermediate vector as a parameter in a local constrained convex optimization problem and updates its estimate to be the solution to this optimization problem. The agent sends its updated estimate to all neighbors. Finally, once the agent has received estimates from all neighbors, it updates an additional parameter to be used in its convex optimization problem in the next iteration.

D-ADMM requires that every agent send its estimate, an N-vector to every neighbor in every iteration. In a single iteration, only one color of agents may send messages at a time. Therefore, one iteration of D-ADMM takes c times as long as one iteration of CB-DIHT, where c is the number of colors.

2) Distributed Subgradient Algorithm: In the distributed subgradient algorithm, each agent p has it's own estimate $x_p^{(k)}$, which is initially 0. Every agent performs a single step of the distributed consensus algorithm to form $u_p^{(k)}$, a weighted average of its and its neighbors estimates. It then computes a projected gradient step using its local objective function and constraints to obtain $x_p^{(k+1)}$, i.e.,

$$x_p^{(k+1)} = \left[u_p^{(k)} - \eta^{(k)} g_p^{(k)} \right]^{+}_{A_p x_p = b_p}.$$

Here $g_p^{(k)}$ is a subgradient of the objective function of agent p, $\frac{1}{P}\|x_p\|_1$, at $x_p^{(k)}$ and $[\cdot]^{+}_{A_p x_p = b_p}$ denotes the projection onto the constraint set $A_p x_p = b_p$. The step size for iteration k is $\eta^{(k)}$. In every iteration of the distributed subgradient algorithm, each agent sends an N-vector along all of its outgoing edges in that iteration.

If the weights used for the consensus steps satisfy Assumption 3 and the step-size sequence $\{\eta^{(k)}\}_{k\geq 0}$ is square-summable but not summable, then, in a static network, this

TABLE I: Recovery problem parameters.

Problem	N	M	P	K	$\lambda_{max}(A^\mathsf{T}A)$
Sparco 902	1000	200	50	3	1
Sparco 7	2560	600	40	20	1
Sparco 11	1024	256	64	32	≈ 2283

algorithm converges to to the optimal solution of (17). The distributed subgradient algorithm has also been shown to converge in time-varying networks where the topology may be state dependent [11]. We note that requirements on the network dynamics specified in Assumption 2 satisfy the conditions required for convergence of the distributed subgradient algorithm.

B. Evaluation Setup

We show evaluation results for three compressed sensing problems from the Sparco toolbox [27]. In Sparco problems 7 and 11, the measurements are exact, and in problem 902, the measurements are noisy. For each problem, we divide the measurements evenly among the agents so that each agent has M/P measurements.

We evaluate each algorithm's performance on five different classes of graphs. For each graph class, we generate five random instances. The results shown in this section are the averages of the five runs over the five instances. The first graph we consider is a Barabasi-Albert (BA) scale free graph [28]. The second and third graphs are Erdös-Rényi (ER) random graphs [29] where each pair of vertices is connected with probability $pr = 0.25$ and probability $pr = 0.75$, respectively. The fourth and fifth graphs are geometric graphs [30] with vertices placed uniformly at random in a unit square. In the fourth graph, two vertices are connected if they are within a distance of $d = 0.5$ of each other, and in the fifth, vertices are connected if they are within a distance of $d = 0.75$. Of these graphs, the BA graph is the least connected, with 128 edges, on average, for $N = 50$ and 171 edges, on average, for $N = 64$. The ER graph with $pr = 0.75$ is the most connected graph, with 992 edges, on average, for $N = 50$ and 1,514 edges, on average, for $N = 64$.

For the simulations in time-varying networks, for each of the graphs described above, we choose ten random subgraphs, ensuring that the union of these subgraphs is the original graph. We cycle through these ten graphs, using one per time step.

We have implemented all algorithms in Matlab and use CVX [31] to solve the local optimization problems in D-ADMM. D-ADMM requires a graph coloring, which we generate using the heuristic from the Matgraph toolbox [32], as is done in [7]. While we include the preprocessing phase in our results for DIHT, we do not include graph coloring preprocessing in our results for D-ADMM. For DIHT and CB-DIHT, in both static and time-varying network experiments, we use $L = 2.01$ for Sparco problems 902 and 7. For Sparco problem 11, we use two values of L for static networks, $L = 4570$ and $L = 500$. For time-varying networks, we use $L = 4570$ and $L = 600$. We note that for problem 11 with $L = 500$ and $L = 600$, $L < \lambda_{max}(A^\mathsf{T}A)$. Therefore, DIHT and CB-DIHT are not guaranteed to converge for these values. However, our evaluations show that for these choices of L, both algorithms converge to an optimal solution.

For the distributed subgradient algorithm, we experimented with different step-sizes $\eta^{(k)} = \frac{1}{k^a}$, where $a \in \{0.51, 0.6, 0.7, 0.8, 0.9, 1\}$. For the most connected graphs, the ER graph with pr=0.75 and the geometric graph with $d = 0.75$, the choice of a with the fastest convergence was 0.8. For the remaining graphs, the fastest convergence as obtained with with $a = 0.6$. For the results below, we use $a = 0.7$, which was the value with the second best performance in terms of convergence for most graphs.

For D-ADMM, we use $\rho = 1$; this value was experimentally shown to be the best choice in the same evaluation setting that we use here [7].

C. Results for Static Networks

For each algorithm, we measure the number of messages sent in order for $\|x_p^{(t)} - \hat{x}\|/\|\hat{x}\|$ to be less than either 10^{-2} or 10^{-2} at every agent. For D-ADMM and the subgradient algorithm $\hat{x}$ is the optimal sparse solution to problem (2). DIHT and CB-DIHT only guarantee convergence to an L-stationary point, and this point may not be the optimal solution to (2). We therefore use the relevant L-stationary point for $\hat{x}$ where applicable. We note that, in most experiments, DIHT and CB-DIHT do converge to the optimal solution. Details of when and how often this occurs are provided below. For each experiment, we ran the simulation until convergence within the desired accuracy or for 2×10^5 iterations, whichever occurred first.

In DIHT, some messages consist of K values and others consists of N values, while in all of the other algorithms, every message consists of N values. To standardize the bandwidth comparison between the algorithms, we assume that only one value is sent per message. Therefore, when an agent sends an N-vector to its neighbor, this requires N messages. We also measure the time required for convergence in a synchronous network where each value is delivered in one clock tick. For all algorithms, we only allow one value to be sent on a link in each direction per clock tick.

The number of values and number of clock ticks for each algorithm for Sparco problems 902, 7 are shown in Tables II, III. For problems 902 and 11, DIHT outperforms all other algorithms in both bandwidth and time on all graph instances. In all cases, DIHT recovers the optimal solution to the problem. DIHT requires two orders of magnitude fewer values and two orders of magnitude fewer time steps than its closest competitor, D-ADMM, to achieve an accuracy of 10^{-2}. It requires at least one order of magnitude fewer values and one order of magnitude fewer time steps than D-ADMM to achieve an accuracy of 10^{-5}. This is true for all classes of graphs.

Both CB-DIHT and the subgradient algorithm require more values and time steps than D-ADMM for these problems. This indicates that these algorithms pay a price for tolerating network dynamics even when the network is static. For an

TABLE II: Signal recovery in a static network for Sparco problem 902 to accuracies of 10^{-2} and 10^{-5}.

(a) Total number of values transmitted to convergence within specified accuracy.

	Accuracy 10^{-2}				**Accuracy** 10^{-5}			
Graph	**DIHT**	**D-ADMM**	**CB-DIHT**	**Subgradient**	**DIHT**	**D-ADMM**	**CB-DIHT**	**Subgradient**
BA	2.31×10^6	2.01×10^7	2.27×10^8	$> 5 \times 10^{10}$	5.80×10^6	2.31×10^7	7.75×10^8	$> 5 \times 10^{10}$
ER (pr= 0.25)	2.31×10^6	6.40×10^7	8.45×10^8	2.40×10^{10}	5.80×10^6	7.31×10^7	3.19×10^9	$> 1 \times 10^{11}$
ER (pr=0.75)	2.31×10^6	3.26×10^8	1.34×10^9	9.49×10^9	5.80×10^6	3.85×10^8	6.75×10^9	$> 3 \times 10^{11}$
Geo (d=0.5)	2.31×10^6	3.29×10^7	1.38×10^9	$\geq 5.33 \times 10^9$	5.80×10^6	3.80×10^7	3.74×10^9	$> 7 \times 10^{10}$
Geo (d=0.75)	2.31×10^6	1.80×10^8	1.65×10^9	1.11×10^{10}	5.80×10^6	2.31×10^7	5.72×10^9	$> 4 \times 10^{11}$

(b) Total number of time steps to convergence within specified accuracy.

	Accuracy 10^{-2}				**Accuracy** 10^{-5}			
Graph	**DIHT**	**D-ADMM**	**CB-DIHT**	**Subgradient**	**DIHT**	**D-ADMM**	**CB-DIHT**	**Subgradient**
BA	2.64×10^5	3.60×10^5	1.02×10^6	$> 4 \times 10^8$	6.63×10^5	3.60×10^5	4.39×10^6	$> 4 \times 10^8$
ER (pr= 0.25)	2.17×10^5	8.39×10^5	1.58×10^6	7.94×10^7	5.44×10^5	8.39×10^5	5.61×10^6	$> 4 \times 10^8$
ER (pr=0.75)	1.79×10^5	3.54×10^6	8.36×10^5	2.91×10^7	4.50×10^5	3.54×10^6	3.91×10^6	$> 4 \times 10^8$
Geo (d=0.5)	2.17×10^5	9.20×10^5	4.95×10^6	$\geq 2.87 \times 10^7$	5.44×10^5	9.20×10^5	1.09×10^7	$> 4 \times 10^8$
Geo (d=0.75)	4.81×10^5	3.34×10^6	1.21×10^6	3.40×10^7	1.21×10^6	3.34×10^6	4.72×10^6	$> 4 \times 10^8$

TABLE III: Signal recovery in a static network for Sparco problem 7 to accuracies of 10^{-2} and 10^{-5}.

(a) Total number of values transmitted.

	Accuracy 10^{-2}				**Accuracy** 10^{-5}			
Graph	**DIHT**	**D-ADMM**	**CB-DIHT**	**Subgradient**	**DIHT**	**D-ADMM**	**CB-DIHT**	**Subgradient**
BA	6.07×10^6	1.14×10^8	1.57×10^9	$> 1 \times 10^{11}$	1.63×10^7	1.17×10^8	5.23×10^9	$> 1 \times 10^{11}$
ER (pr= 0.25)	6.07×10^6	2.52×10^8	3.76×10^9	$> 3 \times 10^{11}$	1.63×10^7	3.07×10^8	1.24×10^{10}	$> 3 \times 10^{11}$
ER (pr=0.75)	6.07×10^6	9.28×10^8	1.16×10^{10}	3.63×10^{10}	1.63×10^7	1.78×10^9	3.80×10^{10}	$> 9 \times 10^{11}$
Geo (d=0.5)	6.07×10^6	1.23×10^8	7.45×10^9	$> 1 \times 10^{11}$	1.63×10^7	1.23×10^8	5.85×10^{10}	$> 1 \times 10^{11}$
Geo (d=0.75)	6.07×10^6	4.53×10^8	7.99×10^9	6.25×10^{10}	1.63×10^7	7.51×10^8	2.65×10^{10}	$> 1 \times 10^{12}$

(b) Total number of time steps.

	Accuracy 10^{-2}				**Accuracy** 10^{-5}			
Graph	**DIHT**	**D-ADMM**	**CB-DIHT**	**Subgradient**	**DIHT**	**D-ADMM**	**CB-DIHT**	**Subgradient**
BA	6.94×10^5	1.09×10^6	6.62×10^6	$> 1 \times 10^9$	1.86×10^6	1.37×10^6	2.13×10^7	$> 1 \times 10^9$
ER (pr= 0.25)	5.70×10^5	2.89×10^6	6.73×10^6	$> 1 \times 10^9$	1.53×10^6	3.52×10^6	2.14×10^7	$> 1 \times 10^9$
ER (pr=0.75)	4.71×10^5	8.56×10^6	6.74×10^6	3.93×10^7	1.26×10^6	1.64×10^7	2.15×10^7	$> 1 \times 10^9$
Geo (d=0.5)	1.26×10^6	2.98×10^6	2.16×10^7	$> 1 \times 10^9$	3.39×10^6	3.89×10^6	4.84×10^7	$> 1 \times 10^9$
Geo (d=0.75)	5.70×10^5	1.46×10^7	6.62×10^6	9.82×10^7	1.53×10^6	2.32×10^7	2.13×10^7	$> 1 \times 10^9$

accuracy of 10^{-5}, the subgradient algorithm did not converge before the maximum number of iterations. The results shown are thus a lower bound on the true number of values and time steps that this algorithm requires for convergence. CB-DIHT converged to the optimal solution in all experiments, outperforming the subgradient algorithm by at least one order of magnitude in bandwidth and time steps in most cases.

The results for Sparco problem 11 are shown in Table IV. Here we only show results for convergence to within 10^{-2}. For DIHT, $L = 4570$ is sufficient to guarantee convergence to an L-stationary point. However, convergence with this L is slower than the convergence of D-ADMM, sometimes requiring up to one order of magnitude more time steps, although DIHT used less bandwidth than D-ADMM. With $L = 4570$, DIHT converged to an L-stationary point that was suboptimal. The choice of $L = 500$ is not sufficient to guarantee convergence under Theorem 3.4. However, in all simulations, DIHT with $L = 500$ converged to the optimal solution. In addition, with the smaller L, DIHT sent one to three orders of magnitude fewer values than D-ADMM and required fewer time steps, also on the order of one to three orders of magnitude. The performance of CB-DIHT is also significantly worse for $L = 4570$ than for $L = 500$, by several orders of magnitude. Additionally, for $L = 4570$, CB-DIHT converged to a suboptimal L-stationary point in all but two graph instances. With $L = 500$, CB-DIHT converged to the optimal solution in all cases. These results indicate that the bound on L in Theorem 4.1 is not tight and that further

TABLE IV: Signal recovery in a static network for Sparco problem 11 to accuracy of 10^{-2}.

(a) Total number of values transmitted.

Graph	DIHT $L = 4570$	DIHT $L = 500$	D-ADMM	CB-DIHT $L = 4570$	CB-DIHT $L = 500$	Subgradient
BA	2.55×10^7	1.53×10^6	8.15×10^7	2.27×10^{10}	3.06×10^8	$> 5 \times 10^{10}$
ER (pr=0.25)	2.55×10^7	1.53×10^6	5.30×10^8	6.60×10^{10}	6.96×10^8	$> 1 \times 10^{11}$
ER (pr=0.75)	2.55×10^7	1.53×10^6	4.83×10^9	4.95×10^{10}	3.66×10^9	1.56×10^{11}
Geo (d=0.5)	2.55×10^7	1.53×10^6	2.26×10^8	4.34×10^{10}	2.54×10^{10}	$> 7 \times 10^{10}$
Geo(d=0.75)	2.55×10^7	1.53×10^6	2.91×10^9	7.73×10^{10}	1.50×10^9	4.02×10^{10}

(b) Total number of time steps.

Graph	DIHT $L = 4570$	DIHT $L = 500$	D-ADMM	CB-DIHT $L = 4570$	CB-DIHT $L = 500$	Subgradient
BA	3.12×10^6	1.46×10^5	9.49×10^5	6.76×10^7	1.27×10^6	$> 4 \times 10^8$
ER (pr=0.25)	2.39×10^6	1.12×10^5	9.49×10^5	6.75×10^7	1.22×10^6	$> 4 \times 10^8$
ER (pr=0.75)	2.08×10^6	9.73×10^4	3.56×10^7	1.68×10^7	1.16×10^6	1.04×10^8
Geo (d=0.5)	5.61×10^6	2.63×10^5	3.21×10^6	7.51×10^7	6.96×10^7	$> 4 \times 10^8$
Geo(d=0.75)	2.55×10^7	1.02×10^5	3.29×10^7	3.66×10^7	1.23×10^6	3.48×10^7

investigation into the convergence conditions for both IHT and its distributed variants is warranted. For experiments with the BA graph, the ER graph with $pr = 0.25$ and the geometric graph with $d = 0.5$, the subgradient algorithm required more than 2×10^5 iterations to converge to within 10^{-2}. Thus, the values in the table are a lower bound on the number of values and number of time steps needed for convergence. As the table shows, CB-DIHT with the larger L outperformed the subgradient method in terms of both time and bandwidth for all but one graph (Geo $d = 0.75$). With the smaller value of L, CB-DIHT always outperformed the subgradient method in both time and bandwidth, usually by at least one order of magnitude.

One interesting thing to note about the results for all problems is that for DIHT, CB-DIHT, and the subgradient algorithm, as the network connectivity increases, both the bandwidth and time needed for convergence decrease. In contrast, in D-ADMM, as network connectivity increases, the algorithm performance gets worse, requiring both more bandwidth and time. In the problem formulation used by D-ADMM, additional constraints are introduced for each edge in the network graph; it is our intuition that these additional constraints lead to a decreased convergence rate. In DIHT, the more connected the network, the less bandwidth and time is needed to compute each sum. CB-DIHT and the subgradient algorithm both employ distributed consensus algorithms. It is well known that consensus algorithms converge more quickly in more connected graphs. We believe that the increase in the convergence rate of the consensus algorithm is carried through to the converge rates of CB-DIHT and the subgradient algorithm.

D. Results for Time-Varying Networks

We compare the performance of CB-DIHT with the distributed subgradient method in time-varying networks. Both algorithms use distributed consensus as a building block; in the subgradient method, agents perform one consensus round per iteration, where agents exchange N-vectors with their neighbors in that round. In CB-DIHT, multiple diffusive consensus rounds are performed for each iteration of DIHT and in each consensus round, agents exchange N-vectors with their neighbors in that round. We compare the performance of CB-DIHT and the subgradient algorithm by counting the number of consensus rounds needed for $\|x_p^{(t)} - \hat{x}\|/\|\hat{x}\|$ to be less than either 10^{-2} or 10^{-5} at every agent, where $\hat{x}$ is as defined in the static network evaluations above.

The results for time-varying networks are shown in Table V. We ran each experiment for a maximum of 3×10^5 consensus rounds. For the subgradient algorithm, every graph instance required more than 3×10^5 consensus rounds to converge to within 10^{-5} of $\hat{x}$. Therefore, we do not show these values in the table. For Sparco problems 902 and 7, CB-DIHT converged to the optimal solution in every instance. In problem 902, CB-DIHT outperformed the subgradient algorithm by as much as two orders of magnitude for an accuracy of 10^{-2}. CB-DIHT required at least one order of magnitude fewer consensus rounds to achieve an accuracy of 10^{-5} in all cases. For Sparco problem 7, CB-DIHT required at least one order of magnitude fewer consensus rounds for both accuracies.

As with the static networks experiments, for CB-DIHT on Sparco problem 11, we use a value of L that is sufficient to guarantee convergence, $L = 4750$, and a smaller value, $L = 600$, that is not sufficient to guarantee convergence, but nevertheless, converges in all experiments. For the larger value of L, CB-DIHT converged to a suboptimal L-stationary point in all but four experiments. For $L = 600$, CB-DIHT always converged to the optimal solution. For both values of L, CB-DIHT required fewer consensus rounds to converge than the subgradient algorithm. This difference is more pronounced with $L = 600$, where CB-DIHT outperformed the subgradient algorithm by at least two orders of magnitude for both accuracies. These results reinforce the need for further investigation

TABLE V: Number of iterations of distributed consensus needed by CB-DIHT and the subgradient algorithm for signal recovery in a time-varying network. For the smaller accuracy, the subgradient algorithm required more than 3×10^5 iterations in every instance.

(a) Sparco problem 902

Graph	Acc. 10^{-2}		Acc. 10^{-5}
	CB-DIHT	**Subgradient**	**CB-DIHT**
BA	2.2×10^3	$> 3 \times 10^5$	7.4×10^3
ER (pr=0.25)	2.5×10^3	$\geq 2.0 \times 10^5$	7.3×10^3
ER (pr=0.75)	1.8×10^3	7.4×10^3	6.0×10^3
Geo (d=0.5)	6.5×10^3	$> 3 \times 10^5$	1.5×10^4
Geo (d=0.75)	1.8×10^3	1.3×10^4	6.0×10^3

(b) Sparco problem 7

Graph	Acc. 10^{-2}		Acc. 10^{-5}
	CB-DIHT	**Subgradient**	**CB-DIHT**
BA	3.0×10^3	$> 3 \times 10^5$	9.3×10^3
ER (pr=0.25)	2.0×10^4	$> 3 \times 10^5$	3.4×10^4
ER (pr=0.75)	3.3×10^3	4.0×10^4	9.8×10^3
Geo (d=0.5)	4.9×10^4	$> 3 \times 10^5$	7.0×10^4
Geo (d.0.75)	3.5×10^3	$\geq 6.8 \times 10^4$	1.0×10^4

(c) Sparco problem 11

Graph	Acc. 10^{-2}			Acc. 10^{-5}	
	CB-DIHT ($L = 4750$)	**CB-DIHT** ($L = 600$)	**Subgradient**	**CB-DIHT** ($L = 600$)	**CB-DIHT** ($L = 4750$)
BA	6.7×10^4	2.3×10^3	$> 3 \times 10^5$	3.4×10^5	3.9×10^3
ER (pr=0.25)	7.1×10^4	8.1×10^2	$> 3 \times 10^5$	1.5×10^5	2.0×10^3
ER (pr=0.75)	5.8×10^4	7.7×10^2	$> 3 \times 10^5$	1.3×10^5	1.9×10^3
Geo (d=0.5)	1.0×10^5	5.7×10^3	$> 3 \times 10^5$	3.1×10^5	8.0×10^3
Geo (d.0.75)	5.2×10^4	8.7×10^2	$> 3 \times 10^5$	1.3×10^5	2.0×10^3

into the relationship between L and the convergence behavior of CB-DIHT.

VI. Conclusion

We have presented two algorithms for in-network, sparse signal recovery based on Iterative Hard Thresholding. We first proposed, DIHT, a distributed implementation of IHT for static networks that combines a novel decomposition of centralized IHT with standard tools from distributed computing. We then proposed an extension of DIHT for time-varying networks. We first showed how centralized IHT can be extended to accommodate inexact computations in each iteration. We then leveraged these new theoretical results to develop CB-DIHT, a version of DIHT that uses a consensus algorithm to execute these inexact computations in a distributed fashion. Our evaluations have shown that, in static networks, DIHT outperforms the best-known distributed compressed sensing in terms of both bandwidth and time by several orders of magnitude. In time-varying networks, CB-DIHT outperforms the distributed subgradient algorithm, the only other algorithm for distributed compressed sensing that accommodates changing network topologies. We also note that, unlike previously proposed algorithms, both DIHT and CB-DIHT can be applied to recovery problems beyond distributed compressed sensing, including recovery from nonlinear measurements.

In future work, we plan to extend our distributed algorithms to support tracking of sparse, time-varying signals. We also plan to explore the application of DIHT and CB-DIHT to problems in the Smart Grid.

Acknowledgment

The work of S. Patterson is funded in part by the Arlene & Arnold Goldstein Center at the Technion Autonomous Systems Program, a Technion fellowship, an Andrew and Erna Finci Viterbi fellowship, and the Lady Davis Fellowship Trust. The work of Y. Eldar is supported in part by the Israel Science Foundation under Grant no. 170/10, in part by the Ollendorf Foundation, and in part by the Intel Collaborative Research Institute for Computational Intelligence (ICRI-CI). The work of I. Keidar is funded in part by the Israel Science Foundation under Grant no. 1549/_11 and the Technion Autonomous Systems Program.

Appendix A
Proofs for Iterative Hard Thresholding with Inexact Gradients

For convenience, we restate some relevant results from [33] and [9].

Lemma A.1 (Descent Lemma): Let f be a continuously differentiable function whose gradient ∇f is Lipschitz continuous over $\mathbb{R}^N$ with constant L_f. Then, for every $L \geq L_f$,

$$f(x) \leq h_L(x, y), \quad \forall x, y \in \mathbb{R}^N,$$

where

$$h_L(x, y) := f(y) + \langle \nabla f(y), x - y \rangle + \tfrac{L}{2}\|x - y\|^2, \ x, y \in \mathbb{R}^N. \tag{18}$$

Lemma A.2 (Lemma 2.2 from [9]): For any $L > 0$, $\hat{x}$ is an L-stationary point of problem (1) if and only if $\|\hat{x}\|_0 \leq K$ and, for $i = 1 \dots N$,

$$|\nabla_i f(\hat{x})| \begin{cases} \leq L\mathcal{M}_K(\hat{x}) & \text{if } \hat{x}_i = 0 \\ = 0 & \text{if } \hat{x}_i \neq 0, \end{cases}$$

where for a given vector v, $\mathcal{M}_K(v)$ returns the absolute value of the K^{th} largest magnitude component of v.

We first derive the following lemma about the relationship between the iterates in k and $k+1$ that are generated by IHT with approximate gradients (analogous to Lemma 2.3 in [9] for IHT with exact gradients).

Lemma A.3: Let $x^{(0)}$ be a K-sparse vector, let $\{x^{(k)}\}_{k\geq 0}$ be the sequence generated by IHT with inexact gradients in (7) with $L > L_f$, and let f satisfy Assumption 1. Then, the following inequality holds for all $k \geq 0$:

$$f(x^{(k)}) - f(x^{(k+1)}) \geq \frac{L-L_f}{2}\|x^{(k)} - x^{(k+1)}\|^2 + \left(x^{(k)} - x^{(k+1)}\right)^{\mathsf{T}} \epsilon^{(k)}. \quad (19)$$

Proof: Let C_s be the set of K-sparse real vectors with N components. The iteration (7) is equivalent to,

$$\begin{aligned} x^{(k+1)} &= \arg\min_{v\in C_s} \left\| v - \left(x^{(k)} - \tfrac{1}{L}\left(\nabla f(x^{(k)}) + \epsilon^{(k)}\right)\right)\right\|^2 \\ &= \arg\min_{v\in C_s} \left\| v - \left(x^{(k)} - \tfrac{1}{L}\nabla f(x^{(k)})\right)\right\|^2 + \frac{2}{L} v^{\mathsf{T}}\epsilon^{(k)} \\ &= \arg\min_{v\in C_s} v^{\mathsf{T}}v - v^{\mathsf{T}}x^{(k)} + \tfrac{2}{L}v^{\mathsf{T}}\nabla f(x^{(k)}) + \tfrac{2}{L}v^{\mathsf{T}}\epsilon^{(k)} \\ &= \arg\min_{v\in C_s} \tfrac{2}{L}(v - x^{(k)})^{\mathsf{T}}\nabla f(x^{(k)}) + \|v - x^{(k)}\|^2 \\ &\qquad + \tfrac{2}{L}v^{\mathsf{T}}\epsilon^{(k)} \\ &= \arg\min_{v\in C_s} h_L(v, x^{(k)}) \ + \ v^{\mathsf{T}}\epsilon^{(k)}, \end{aligned}$$

where h_L is as defined in (18). The above implies that,

$$h_L(x^{(k+1)}, x^{(k)}) + (x^{(k+1)} - x^{(k)})^{\mathsf{T}}\epsilon^{(k)} \leq h_L(x^{(k)}, x^{(k)}) \leq f(x^{(k)}). \quad (20)$$

By Lemma A.1, we have

$$\begin{aligned} f(x^{(k)}) - f(x^{(k+1)}) &\geq f(x^{(k)}) - h_{L(f)}(x^{(k+1)}, x^{(k)}) \\ &\geq f(x^{(k)}) - h_L(x^{(k+1)}, x^{(k)}) \\ &\quad + \tfrac{L-L(f)}{2}\|x^{(k)} - x^{(k+1)}\|^2. \quad (21) \end{aligned}$$

Combining (20) and (21), we obtain the result in (19). ∎

Using this lemma, we can establish the convergence of the sequence $\{\|x^{(k)} - x^{(k+1)}\|^2\}_{k\geq 0}$, provided the sequence of error terms $\{\epsilon^{(k)}\}_{t\geq 0}$ is square-summable.

Lemma A.4: Let $x^{(0)}$ be a K-sparse vector, and let $\{x^{(k)}\}_{k\geq 0}$ be the sequence generated by IHT with inexact gradients in (7) with constant step size $L > L_f$. Let the sequence $\{\epsilon^{(k)}\}_{k\geq 0}$ be such that $\sum_{k=0}^{\infty}\|\epsilon^{(k)}\|^2 = E < \infty$. Then,

1) There exists $D < \infty$ such that

$$\sum_{k=0}^{T}\|x^{(k)} - x^{(k+1)}\|^2 \leq D, \text{ for all } T \geq 0.$$

2) $\lim_{k\to\infty}\|x^{(k)} - x^{(k+1)}\|^2 = 0$.

Proof:

To prove the first part of the lemma, we show that the sequence

$$\left\{\sum_{k=0}^{T}\|x^{(k)} - x^{(k+1)}\|^2\right\}_{T\geq 0}$$

is bounded.

Consider the sum over time of $f(x^{(k)}) - f(x^{(k+1)})$. We can bound this sum as follows,

$$\begin{aligned} \sum_{k=0}^{T}\left(f(x^{(k)}) - f(x^{(k+1)})\right) &= f(x^{(0)}) - f(x^{(T+1)}) \\ &\leq f(x^{(0)}) - B, \end{aligned}$$

where the last inequality follows from the fact that f is lower bounded by a constant B (see Assumption 1). Note that this bound holds for all $T \geq 0$.

Define $A := f(x^{(0)}) - B$, and note that since $f(x^{(0)})$ is finite, A is also finite. By Lemma A.3, we have the following for all $T \geq 0$,

$$A \geq \sum_{k=0}^{T}\left(\tfrac{L-L_f}{2}\|x^{(k)} - x^{(k+1)}\|^2 + \left(x^{(k)} - x^{(k+1)}\right)^{\mathsf{T}}\epsilon^{(k)}\right) \quad (22)$$

$$\geq \sum_{k=0}^{T}\left(\tfrac{L-L_f}{2}\|x^{(k)} - x^{(k+1)}\|^2\right) - \sum_{k=0}^{T}\left(\|x^{(k)} - x^{(k+1)}\|\|\epsilon^{(k)}\|\right) \quad (23)$$

$$\geq \sum_{k=0}^{T}\left(\tfrac{L-L_f}{2}\|x^{(k)} - x^{(k+1)}\|^2\right) - \sqrt{E}\sqrt{\sum_{k=0}^{T}\|x^{(k)} - x^{(k+1)}\|^2}, \quad (24)$$

where (23) follows from (22) by the Cauchy-Schwarz inequality, and (24) follows from (23), again by Cauchy-Schwarz and by the assumption on the square-summability of the error.

For clarity of notation, let $\beta := \frac{L-L_f}{2}$, $C := \sqrt{E}$, and $D := \sqrt{\sum_{t=0}^{T}\|x^{(k)} - x^{(k+1)}\|^2}$. We can then rewrite (24) as,

$$-\beta D^2 + CD + A \geq 0. \quad (25)$$

Our goal is to show that the sum D is bounded (for all $T \geq 0$). Equivalently, we must show that every $D >= 0$ that satisfies (25) is bounded. The non-negative values of D that satisfy (25) that satisfy are such that,

$$0 \leq D \leq \frac{C + \sqrt{C^2 + 4\beta A}}{2\beta}.$$

Since A is finite, the sum D is bounded for all T, thus proving part one of the lemma.

We now show that the sequence $\{\sum_{t=0}^{T}\|x^{(k)} - x^{(k+1)}\|^2\}_{T\geq 0}$ converges, which implies part two of the lemma (see [34], Theorem 3.23). To this end, we must show that the sequence is monotonically increasing and bounded. We have already established that it is bounded. Monotonicity is easily established by,

$$\begin{aligned} \sum_{k=0}^{T}\|x^{(k)} - x^{(k+1)}\|^2 &= \sum_{k=0}^{T-1}\|x^{(k)} - x^{(k+1)}\|^2 \\ &\quad + \|x^{(T)} - x^{(T+1)}\|^2 \\ &\geq \sum_{k=0}^{T-1}\|x^{(k)} - x^{(k+1)}\|^2, \end{aligned}$$

completing the proof. ∎

We now prove the main results of Section IV-A.

Theorem 4.1 (restated) *Let f be lower-bounded and let ∇f be Lipschitz-continuous with constant L_f. Let $\{x^{(k)}\}_{k\geq 0}$ be the sequence generated by algorithm (7) with $L > L_f$ and with a sequence $\{\epsilon^{(k)}\}_{k\geq 0}$ satisfying $\sum_{k=0}^{\infty}\|\epsilon^{(k)}\|^2 < \infty$. Then, any accumulation point of $\{x^{(k)}\}_{k\geq 0}$ is an L-stationary point.*

Proof: Let $\hat{x}$ be an accumulation point of the sequence $\{x^{(k)}\}_{k\geq 0}$. Then, there exists a subsequence $\{x^{(k_r)}\}_{r\geq 0}$ that converges to $\hat{x}$. The sequences $\left\{f\left(x^{(k_r)}\right)\right\}_{r\geq 0}$ and $\left\{f\left(x^{(k_r+1)}\right)\right\}_{r\geq 0}$ converge to the same limit f^*, and therefore

$$\lim_{r\to\infty} f\left(x^{(k_r)}\right) - f\left(x^{(k_r+1)}\right) = 0.$$

Combining the above with Lemma A.3, we have $\lim_{r\to\infty} x^{(k_r+1)} = \hat{x}$.

Consider the non-zero components of $\hat{x}$, i.e., components with $\hat{x}_i \neq 0$. Since both $x^{(k_r)}$ and $x^{(k_r+1)}$ converge to $\hat{x}$ (as $k\to\infty$), there exists an R such that

$$x_i^{(k_r)}, x_i^{(k_r+1)} \neq 0, \text{ for all } r \geq R.$$

Therefore, for $r \geq R$,

$$x_i^{(k_r+1)} = x_i^{(k_r)} - \tfrac{1}{L}\left(\nabla_i f\left(x^{(k_r)}\right) + \epsilon^{(k_r)}\right). \tag{26}$$

Since, $\sum_{k=0}^{\infty}\|\epsilon^{(k)}\|^2$ is bounded, we have $\lim_{k\to\infty}\epsilon^{(k)} = 0$. Thus, taking r to ∞ in (26), we obtain that $\nabla_i f(\hat{x}) = 0$.

Now consider the zero components of $\hat{x}$, i.e. components with $\hat{x}_i = 0$. If there exist an infinite number of indices k_r for which $x_i^{(k_r+1)} \neq 0$, then, as before,

$$x_i^{(k_r+1)} = x_i^{(k_r)} - \tfrac{1}{L}\left(\nabla_i f\left(x^{(k_r)}\right) + \epsilon^{(k_r)}\right),$$

which implies $\nabla_i f(\hat{x}) = 0$, and thus $\|\nabla_i f(\hat{x})\| \leq L\mathcal{M}_K(\hat{x})$. If there exists a $Q \geq 0$ such that for all $r \geq Q$, $x_i^{(k_r+1)} = 0$, then,

$$\begin{aligned}&\left|x_i^{(k_r)} - \tfrac{1}{L}\left(\nabla_i f(x^{(k_r)}) + \epsilon^{(k_r)}\right)\right| \\ &\quad\leq\ \mathcal{M}_K\left(x^{(k_r)} - \tfrac{1}{L}\nabla f\left(x^{(k_r)}\right)\right) = \mathcal{M}_K\left(x^{(k_r+1)}\right).\end{aligned}$$

Taking r to infinity, we obtain

$$\left|\hat{x} - \tfrac{1}{L}\nabla_i f(\hat{x})\right| \leq \mathcal{M}_K(\hat{x}),$$

or equivalently, $|\nabla_i f(\hat{x})| \leq L\mathcal{M}_K(\hat{x})$. Therefore, by Lemma A.2, $\hat{x}$ is an L-stationary point, thus proving the theorem. ∎

Theorem 4.2 (restated) *Let $f(x) = \|Ax-b\|^2$, with A satisfying the K-regularity property. Let $\{x^{(k)}\}_{k\geq 0}$ be the sequence generated by (7) with $L > 2\lambda_{max}(A^{\mathsf{T}}A)$ and with a sequence $\{\epsilon^{(k)}\}_{t\geq 0}$ satisfying $\sum_{k=0}^{\infty}\|\epsilon^{(k)}\|^2 < \infty$. Then, the sequence $\{x^{(k)}\}_{k\geq 0}$ converges to an L-stationary point.*

Proof: First, we show that the sequence $\{x^{(k)}\}_{k\geq 0}$ is bounded. Applying Lemma A.3, we have

$$\begin{aligned}\sum_{s=0}^{k-1} f(x^{(s)}) - f(x^{(s+1)}) &\geq \sum_{s=0}^{k-1}\tfrac{L-L_f}{2}\|x^{(s)}-x^{(s+1)}\|^2 \\ &\quad - \sum_{s=0}^{t-1}\|x^{(s)}-x^{(s+1)}\|\|\epsilon^{(s)}\| \qquad (27)\\ f\left(x^{(0)}\right) - f\left(x^{(k)}\right) &\geq \sum_{s=0}^{t-1}\tfrac{L-L_f}{2}\|x^{(s)}-x^{(s+1)}\|^2 \\ &\quad - \sqrt{\sum_{s=0}^{k-1}\|\epsilon^{(s)}\|^2)}\sqrt{\sum_{s=0}^{k-1}\|x^{(s)}-x^{(s+1)}\|^2} \qquad (28)\end{aligned}$$

By the assumption on the square summability of the error terms, there exists $E < \infty$ such that

$$\sqrt{E} = \sqrt{\sum_{s=0}^{k-1}\|\epsilon^{(s)}\|^2}.$$

We also define $0 \leq F < \infty$ such that

$$\sqrt{F} = \sqrt{\sum_{s=0}^{k-1}\|x^{(s)}-x^{(s+1)}\|^2}.$$

Note that, by Lemma A.4, such an F exists. With these definitions, we arrive at the following inequality for $k \geq 0$,

$$f(x^{(0)}) + \sqrt{E}\sqrt{F} \geq f(x^{(k)}).$$

Define the level set,

$$T := \left\{x \in \mathbb{R}^N : f(x) \leq f(x^{(0)}) + \sqrt{E}\sqrt{F}\right\},$$

and note that the sequence $\{x^{(k)}\}_{k\geq 0}$ is contained in this set. For $f(x) = \|Ax-b\|_2^2$, if x is K-sparse and A is K-regular, then the set T is bounded (see [9], Theorem 3.2). Thus the sequence $\{x^{(k)}\}_{k\geq 0}$ is bounded.

The remainder of this proof is nearly identical to that of Theorem 3.2 in [9]. We include it here for completeness. The boundedness of $\{x^{(k)}\}_{k\geq 0}$ implies that there exists a subsequence $\{x^{(k_j)}\}_{j\geq 0}$ that converges to an L-stationary point x^*. By Lemma 2.1 in [9], there are only a finite number of L-stationary points. Assume that the sequence $\{x^{(k)}\}_{k\geq 0}$ does not converge to x^*. This means that,

$$\exists\epsilon_1 > 0, \text{ such that } \forall J \geq 0, \exists j > J \text{ with } \|x^{(j)} - x^*\| \geq \epsilon_1.$$

Define $\epsilon_2 > 0$ to be less than the minimum distance between all pairs of L-stationary points, and define $\epsilon := \min(\epsilon_1, \epsilon_2)$. Without loss of generality, assume that $\{x^{(k_j)}\}_{j\geq 0}$ satisfies $\|x^{k_j} - x^*\| \leq \epsilon$ for all $j \geq 0$, and define the sequence $\{x^{(t_j)}\}_{j\geq 0}$ with,

$$t_j = \max\{l : \|x^i - x^*\| \leq \epsilon, i = k_j, k_j+1, \ldots, l\}.$$

We know that each t_j is well-defined by the assumption that $\{x^{(k)}\}_{k\geq 0}$ does not converge to x^*. Given the definition of $x^{(t_j)}$ and the fact that $\|x^{(k)} - x^{(k+1)}\| \to 0$ (by Lemma A.4), there exists a subsequence of $x^{(t_j)}$ that converges to a point z^* with $\|x^* - z^*\| \geq \epsilon$. The existence of this

subsequence contradicts the assumption that ϵ was chosen so every accumulation point $y \neq x^*$ is such that $\|x^* - y\| > \epsilon$. Therefore, the sequence $\{x^{(k)}\}_{k\geq 0}$ converges to x^*. ■

APPENDIX B
PSEUDOCODE FOR OTHER RECOVERY ALGORITHMS

The pseudocode for D-ADMM is given in Algorithm 6. The pseudocode for the distributed subgradient algorithm is given in Algorithm 7.

Algorithm 6: D-ADMM for agent p with color c. Here D_p denotes the node degree of agent p.

initialize

$x_p^{(0)} \leftarrow 0$

$\gamma_p^{(0)} \leftarrow 0$

$k \leftarrow 0$

while TRUE **do**

on receive $x_q^{(k+1)}$ from neighbors with lower colors

$$u_p^{(k)} \leftarrow \gamma_p^{(k)} - \rho \sum_{\substack{q\in\mathcal{N}p \\ col(q)<c}} x_q^{(k+1)} - \rho \sum_{\substack{q\in\mathcal{N}p \\ col(q)>c}} x_q^{(k)}$$

$$x_p^{(k+1)} \leftarrow \arg\min_{x_p} \tfrac{1}{P}\|x_p\|_1 + {u_p^{(k)}}^{\mathsf{T}} x_p + \tfrac{D_p}{2}\|x_p\|_2^2$$

$$\text{subject to } A_p x_p = b_p$$

send $x_p^{(k+1)}$ to all neighbors

on receive $x_q^{(k+1)}$ from all neighbors

$$\gamma_p^{(k+1)} \leftarrow \gamma_p^{(k)} + \rho \sum_{q\in\mathcal{N}p} \left(x_p^{(x+1)} - x_q^{(x+1)}\right)$$

$k \leftarrow k + 1$

Algorithm 7: Distributed subgradient algorithm.

initialize

$x_p^{(0)} \leftarrow 0$

$k \leftarrow 0$

while TRUE **do**

$u_p^{(k)} \leftarrow \sum_{q=1}^{P} [W(k)]_{pq} x_q^{(k)}$

$g_p^{(k)} \leftarrow$ subgradient of $\|x_p\|_1$ at $x_p^{(k)}$

$$x_p^{(k+1)} \leftarrow \left[u_p^{(k)} - \eta^{(k)} g_p^{(k)}\right]^{+}_{A_p x_p = b_p}$$

REFERENCES

[1] Y. C. Eldar and G. Kutyniok, *Compressed Sensing: Theory and Applications*. Cambridge University Press, 2012.

[2] J. Meng, H. Li, , and Z. Han, "Sparse event detection in wireless sensor networks using compressive sensing," in *Proc 43rd Ann. Conf. Information Sciences and Systems*, 2009.

[3] Z. Li, Y. Zhu, H. Zhu, and M. Li, "Compressive sensing approach to urban traffic sensing," in *Proc. 31st Int. Conf. Distributed Computing Systems*, 2011, pp. 889–898.

[4] X. Yu, H. Zhao, L. Zhang, S. Wu, B. Krishnamachari, and V. O. K. Li, "Cooperative sensing and compression in vehicular sensor networks for urban monitoring," in *2010 IEEE Int. Conf. on Communications*, 2010, pp. 1–5.

[5] J. A. Bazerque and G. B. Giannakis, "Distributed spectrum sensing for cognitive radio networks by exploiting sparsity," *IEEE Trans. Sig. Proc.*, vol. 58, no. 3, pp. 1847–1862, 2010.

[6] J. Mota, J. Xavier, P. Aguiar, and M. Püschel, "Basis pursuit in sensor networks," in *Proc. IEEE Int. Conf. on Acoust., Speech, and Sig. Proc. (ICASSP)*, 2011, pp. 2916–2919.

[7] ——, "Distributed basis pursuit," *IEEE Trans. Sig. Proc.*, vol. 60, no. 4, pp. 1942–1956, Apr 2012.

[8] T. Blumensath and M. E. Davies, "Iterative hard thresholding for compressed sensing," *Applied and Computational Harmonic Analysis*, vol. 27, no. 3, pp. 265–274, 2009.

[9] A. Beck and Y. C. Eldar, "Sparsity constrained nonlinear optimization: Optimality conditions and algorithms," *CoRR*, vol. abs/1203.4580, 2012.

[10] J. N. Tsitsiklis, "Problems in decentralized decision making and computation," Ph.D. thesis, Massachusetts Institute of Technology, 1984.

[11] I. Lobel, A. E. Ozdaglar, and D. Feijer, "Distributed multi-agent optimization with state-dependent communication," *Math. Program.*, vol. 129, no. 2, pp. 255–284, 2011.

[12] J. Mota, J. Xavier, P. Aguiar, and M. Püschel, "Distributed algorithms for basis pursuit," in *Proc. 2nd Int. Workshop Signal Process. Adapt. Sparse Structured Represent.*, 2009.

[13] A. Nedic and A. Ozdaglar, "Distributed subgradient methods for multi-agent optimization," *IEEE Trans. Autom. Contr.*, vol. 54, no. 1, pp. 48–61, 2009.

[14] C. Ravazzi, S. Fosson, and E. Magli, "Distributed soft thresholding for sparse signal recovery," *CoRR*, vol. abs/1301.2130, 2013.

[15] A. I. Chen and A. E. Ozdaglar, "A fast distributed proximal-gradient method," in *Proc. of Allerton Conference on Communication, Control, and Computing*, 2012.

[16] M. Schmidt, N. Le Roux, and F. Bach, "Convergence rates of inexact proximal-gradient methods for convex optimization," in *25th Ann. Conf. Neural Inf. Proc. Sys.*, 2011.

[17] T. Blumensath and M. E. Davies, "Iterative thresholding for sparse approximations," *J. Fourier Analysis and Applications*, vol. 14, no. 5, pp. 629–654, Dec 2008.

[18] T. Blumensath, "Compressed sensing with nonlinear observations and related nonlinear optimisation problems," *IEEE Trans. Information Theory*, vol. 59, no. 6, pp. 3466–3474, 2013.

[19] S. Patterson, Y. C. Eldar, and I. Keidar, "Distributed sparse signal recovery in sensor networks," in *Proc. IEEE Int. Conf. on Acoust., Speech, and Sig. Proc.*, 2013.

[20] N. Lynch, *Distributed Algorithms*. USA: Morgan Kaufmann Publishers, Inc., 1996.

[21] V. D. Blondel, J. M. Hendrickx, A. Olshevsky, and J. N. Tsitsiklis, "Convergence in multiagent coordination, consensus, and flocking," in *Proc. Joint 44th IEEE Conference on Decision and Control and European Control Conference*, 2005, pp. 2996–3000.

[22] B. K. Natarajan, "Sparse approximate solutions to linear systems," *SIAM J. Comput.*, vol. 24, no. 2, pp. 227–234, Apr 1995.

[23] E. J. Candès, "Compressive sampling," in *Proc. Int. Congr. Math.*, 2006, pp. 1433–1452.

[24] A. Segall, "Distributed network protocols," *IEEE Trans. Inf. Theory*, vol. 29, no. 1, pp. 23–35, 1983.

[25] H. Attiya and J. Welch, *Distributed Computing: Fundamentals, Simulations, and Advanced Topics*, 2nd ed. Wiley-Interscience, 2004, vol. 19.

[26] S. Chen, D. Donoho, and M. Saunders, "Atomic decomposition by basis pursuit," *SIAM J. Sci. Comput.*, vol. 20, no. 1, pp. 33–61, 1998.

[27] E. v. Berg, M. P. Friedlander, G. Hennenfent, F. Herrmann, R. Saab, and Ö. Yılmaz, "Sparco: A testing framework for sparse reconstruction," Dept. Computer Science, University of British Columbia, Vancouver, Tech. Rep. TR-2007-20, October 2007.

[28] A. Barabási and R. Albert, "Emergence of scaling in random networks," *Science*, vol. 286, pp. 409–410, 1999.

[29] P. Erdös and A. Rényi, "On random graphs I." *Publicationes Mathematicae*, vol. 6, pp. 290–297, 1959.

[30] M. Penrose, *Random Geometric Graphs*. Oxford, U.K.: Oxford University Press, 2004.

[31] M. Grant and S. Boyd, "CVX: Matlab software for disciplined convex programming, version 1.21," http://cvxr.com/cvx, Apr 2011.

[32] E. R. Scheinerman, "Matgraph: A matlab toolbox for graph theory," 2012, online: http://www.ams.jhu.edu/~ers/matgraph/matgraph.pdf.

[33] D. P. Bertsekas, *Nonlinear Programming*. Belmont, MA: Athena Scientific, 1999.

[34] W. Rudin, *Principles of Mathematical Analysis*. McGraw-Hill, Inc., 1976.