Distributed Compressed Sensing For Static and Time-Varying Networks

Stacy Patterson *Member, IEEE,* Yonina C. Eldar *Fellow, IEEE,* 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, sparse recovery**

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 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.

Since the measurements are already distributed throughout the network, it is desirable to perform the signal recovery within the network itself. Distributed solutions for compressed sensing have begun to receive attention lately [5], [6], [7], [8]. Although these algorithms converge to a 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 compressed sensing that is based on *Iterative Hard Thresholding* (IHT) [9]. 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 [10] 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 also be applied to these settings.

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 while providing the same recovery guarantees as the original IHT formulation. We then leverage these new theoretical results

S. Patterson is with the Department of Computer Science, Rensselaer Polytechnic Institute, Troy, NY 12180 USA (email: sep@cs.rpi.edu).

Y. C. Eldar, and I. Keidar are with the Department of Electrical Engineering, Technion - Israel Institute of Technology, Haifa 32000 Israel (email {yonina, idish}@ee.technion.ac.il).

The work of Y. C. Eldar is funded in part by the Intel Collaborative Research Institute for Computational Intelligence (ICRI-CI). The work of I. Keidar is funded in part by the ICRI-CI, the Technion Autonomous Systems Program, and the Israeli Science Foundation.

to develop a version of DIHT that uses multiple rounds of a distributed consensus algorithm [11] to execute each inexact global computation. We call this algorithm *consensus-based DIHT*, or CB-DIHT. Evaluations show that CB-DIHT requires several orders of magnitude less time and bandwidth than the best-known, previously proposed solution.

A. Related Work

Several recent works have proposed distributed algorithms that can apply to a basis pursuit formulation of the distributed compressed sensing problem. In these, the signal is recovered by solving a convex formulation of the original recovery problem. These distributed methods can be divided into two classes: double-looped algorithms and single-looped algorithms.

The double looped techniques [12], [6], [13] 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. With the exception of [13], these algorithms all require a static network. In single-looped methods [5], [7], [8], 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. A recent work [7] presented an experimental evaluation of these methods and demonstrated that the single-looped algorithm D-ADMM [7], [8] outperformed the other algorithms. While D-ADMM uses only local communication, each agent must send its entire estimate vector to every neighbor in every iteration. This vector may not be sparse for many iterations, and therefore, bandwidth usage can be high. Furthermore, the convergence time increases as the network connectivity increases, whereas the convergence rates of DIHT and CB-DIHT improve with increased network connectivity. We note that the convergence of D-ADMM has only been established theoretically for bipartite graphs, but experiments have demonstrated convergence in general graphs.

The distributed subgradient algorithm [14], [15] 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 with its neighbors and then performs a local projected subgradient step. The algorithm converges in static and time-varying networks, though the convergence rate can be slow. Simulations have shown that, in a time-varying graph, the distributed subgradient method converges more quickly than the double looped algorithm in [13].

The work by Ravazzi et al. [16] proposes a distributed 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, static graph.

Our approach for CB-DIHT was inspired by recent work on a distributed proximal gradient algorithm [17]. 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 each inexact computation. This work differs from ours in that the convergence of the inexact centralized proximal gradient method had already been established [18], whereas no such prior analysis exists for centralized IHT. Furthermore, the distributed proximal gradient algorithm depends on assumptions that are not compatible with standard compressed sensing formulations such as the one in this paper.

The convergence of centralized IHT was established in [19], and its application to compressed sensing was studied in [9]. Recently, Beck and Eldar adapted IHT to signal recovery for more general nonlinear objective functions and provided theoretical guarantees on signal recovery in this setting [10]. We leverage this work in our distributed algorithms. A variation on IHT for nonlinear measurements was also proposed in [20]. This work uses a Taylor series approximation for the gradient step in each iteration rather than an exact gradient as in [10].

Finally, we note that, in a related work [21], we present an extension to DIHT for static networks that can further reduce bandwidth for problems that require many rounds to converge.

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 provides numerical results demonstrating the performance of DIHT and CB-DIHT. A discussion on the fault tolerance and recovery guarantees of distributed compressed sensing algorithms is given in Section VI. Concluding remarks are provided in Section VII.

II. Problem Formulation

We consider a network of P agents. The agents may be 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. This agent can be chosen using a variety of well-known distributed algorithms (see [22]).

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 x from their collective measurements using only communication between neighbors in the network. To recover x, the agents attempt to 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 psuedo-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.

The following assumption is made throughout the paper.

Assumption 0: Agent 1 knows the sparsity parameter K of the signal x to be estimated.

We note that this assumption is made only for convenience. In practice, IHT can be implemented without this knowledge by only keeping the elements above a threshold.

We also make the following assumptions about the loss functions.

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

(a) There exists a $B_p \in \mathbb{R}$ such that for all $x \in \mathbb{R}^N$, $f_p(x) \geq B_p$.

(b) The gradient ∇f_p is Lipschitz continuous over $\mathbb{R}^N$ with Lipschitz constant L_{f_p}, i.e.,
$$\|\nabla f_p(x) - \nabla f_p(y)\|_2 \leq L_{f_p}\|x-y\|_2, \quad \forall x, y \in \mathbb{R}^N.$$
Note that this implies that $f(x) = \sum_{p=1}^{P} f_p(x)$ is also Lipschitz continuous over $\mathbb{R}^N$.

(c) There exists $G_p, H_p \in \mathbb{R}$ such that for all $x \in \mathbb{R}^N$,
$$\|\nabla f_p(x)\|_2 \leq G_p\|x\|_2 + H_p.$$

(d) Every agent p knows its L_{f_p}, and agent 1 knows an upper bound on the Lipschitz constant L_f for $f(x) = \sum_{p=1}^{P} f_p(x)$.

The agents do not know B_p, G_p, nor H_p.

If agent 1 does not know an upper bound on L_f, then it can learn one using a distributed algorithm. One such upper bound is $\sum_{p=1}^{P} L_{f_p}$. Distributed algorithms for computing an upper bound for L_f are described in Appendix A.

As a specific example problem, we consider compressed sensing [1]. Here, each agent p has M_p linear measurements of x taken using its sensing matrix $A_p \in \mathbb{R}^{M_p \cdot N}$. 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,
$$\text{minimize } \|Ax - b\|_2^2 \quad \text{subject to } \quad \|x\|_0 \leq K. \tag{2}$$

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

Static Network Model: We model the network by an undirected, 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 [23], [14], [17].

Assumption 2: The sequence of graphs $(V, E^{(t)})$, $t = 0, 1, 2, \ldots$, 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 $(q, p) \in E^{(\infty)}$, then $(q, p) \in E^{(t)} \cup E^{(t+1)} \cup \cdots \cup E^{(t+C-1)}$, 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 $\overline{C} = (P-1)C$ time steps is a strongly connected graph. The agents do not know the value of C.

Our goal, 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 networks.

III. Distributed Algorithm for Static Networks

Problem (1) is known to be NP-Hard in general [24]. However, for suitable loss functions, efficient centralized algorithms exist. Our distributed recovery algorithm is based on Iterative Hard Thresholding [19], [9]. 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 $f : \mathbb{R}^N \to \mathbb{R}$ 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 [19], [9], [10] is a gradient-like, centralized algorithm that recovers x^* by solving the optimization problem,
$$\text{minimize } f(x) \quad \text{subject to } \quad \|x\|_0 \leq K. \tag{3}$$

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 arbitrary K-sparse vector $x^{(0)}$. In each iteration, a gradient-step is performed, followed by application of the thresholding operator. This iteration is given by,
$$x^{(k+1)} = \mathcal{T}_K\left(x^{(k)} - \tfrac{1}{L}\nabla f(x^{(k)})\right), \tag{4}$$
where $L > L_f$ is a constant.

The loss function f 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 [10] 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 Thm. 2.2 in [10]).

With this definition, we can now state the relevant convergence result for IHT with a general nonlinear objective.

Theorem 3.1 (Thm. 3.1 in [10]): 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) .
We note that this theorem does not guarantee that IHT will converge to an L-stationary point; it only guarantees that if the algorithm converges, the accumulation point is an L-stationary point. More details on the convergence behavior of IHT for nonlinear objectives can be found in [10].

For the compressed sensing problem, a stronger result has been shown.

Theorem 3.2 (Thm. 5 in [9]): 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 [25] 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 $z_p^{(k)}$. 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 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,

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

Algorithm 1: DIHT

```
initialize
    $k \leftarrow 0$
    $x_1^{(0)} \leftarrow x_{init}$
Algorithm executed by agent 1.
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 executed by agent $p \neq 1$.
on 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$(p)$
    $sum_p^{(k)} \leftarrow \left(\sum_{q \in \text{children}(p)} sum_q^{(k)}\right) + z_p^{(k)}$
    Send $sum_p^{(k)}$ to parent
    $k \leftarrow k + 1$
```

To find G, first, the agents compute the sum of their intermediate vectors using a well-known distributed algorithm called *broadcast/convergecast* [26] (described below). This sum is then used to complete the gradient step, followed by application of the threshold operator. The combination of the local computation (5) and the global computation (6) is equivalent to one iteration of centralized IHT in (4).

We now describe DIHT. The agents first create a breadth-first spanning tree over the network, rooted at agent 1, using a distributed algorithm (see [27] for details). This 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. The creation of this tree can be done as a pre-processing step, but in practice, the tree creation is usually done in conjunction with the first broadcast.

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 $z_p^{(k)}$ 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. The process is then repeated to obtain the next iterate. Pseudocode for DIHT is given in Algorithm 1.

C. Algorithm Analysis

DIHT requires $O(N)$ storage at each agent. In every iteration, every agent computes the gradient of its local loss function. For 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. An iteration of DIHT consists of a broadcast in which a K-sparse vector is sent down the tree and a convergecast in which the intermediate N-vectors are aggregated up the tree. As the tree has $P-1$ edges, $2(P-1)$ total messages are sent per iteration. For the broadcast, each agent sends at most $D_p - 1$ messages, where D_p is the node degree in the original graph. For the convergecast, the agent sends a single message to its parent in the tree, for a total of D_p messages per agent per iteration. While agent 1 plays a unique role in the global computation, it performs the same number and types of computations as every other agent, with the exception of the thresholding operation which requires a single scan of the sum vector. One approach to find the K largest magnitude values in a single scan is for agent 1 to keep a priority queue, initially containing the first K components of the sum vector. Starting with component $K+1$, the agent checks each remaining component in the sum vector against the smallest entry in the queue. If the component is larger, then the component is added to the queue and the smallest entry is removed. Each enqueue or dequeue operation has $\log(K)$ time complexity, and checking the smallest entry can be done in constant time. Thus, the running time of this approach is $O(N\log(K))$.

The estimates $x_p^{(k)}$, $p = 1 \dots P$, are equivalent to each other in all iterations, and they evolve exactly as $x^{(k)}$ in (6). Thus, DIHT provides the same 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 $L > L_f$ 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 a sum in DIHT 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 perform the global computation in finite time using *any* algorithm. This is because, without this knowledge, an agent cannot determine when it has received the information it needs (from all other agents) to compute the sum in (6).

In our extended DIHT algorithm, we use a distributed consensus algorithm [11] 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 time-varying networks, distributed consensus converges to the average of the agents' initial values [23]. After a finite number of iterations, the agents 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 centralized IHT.

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 ∇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 then leverage these new theoretical results to develop a consensus-based distributed IHT algorithm for time-varying networks.

A. IHT with Inexact Gradients

IHT with inexact gradients is identical to IHT in (4), except that, in each iteration the gradient is computed approximately. The iteration is thus given by,

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

Here, $\epsilon^{(k)} \in \mathbb{R}^N$ is 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. Proofs are deferred to Appendix B. Our first theorem (analogous to Theorem 3.1 in [10]) 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 (7) with $L > L_f$ and with a sequence $\{\epsilon^{(k)}\}_{k\geq 0}$ satisfying $\sum_{k=0}^{\infty}\|\epsilon^{(k)}\|_2^2 < \infty$. 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 [10]). Let $f(x) = \|Ax - b\|_2^2$. The spark of A, denoted spark(A) is the smallest number of columns of A that are linearly dependent. If spark$(A) > K$, then algorithm (7) converges to an L-stationary point.

Theorem 4.2: Let $f(x) = \|Ax - b\|_2^2$, with spark(A) $> K$. 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 $\sum_{k=0}^{\infty}\|\epsilon^{(k)}\|_2^2 < \infty$. Then, the sequence $\{x^{(k)}\}_{k\geq 0}$ converges to an L-stationary point.

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,

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

where $w_{pq}^{(t)}$ is the weight that agent p assigns to the value at agent q. Under appropriate assumptions about the weights and the network connectivity over time (e.g., Assumption 2), the agents' vectors converge to $v_{av} = \frac{1}{P}\sum_{p=1}^{P} v_p^{(0)}$ [23].

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, 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 in 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 step, an agent may be *initiated*, meaning it is participating in the consensus algorithm, 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 in that time step, $i.e.$, it sends messages to all agents p such that $(p, 1) \in E^{(0)}$. Upon receipt of this message, an agent is initiated. In time step 1, the initiated agents begin the consensus algorithm specified by (8) over the active links that are present in that time step. If agent p is initiated in time step t, in all steps $T > t$, it sends INITIATE messages over any adjacent, inactive links, thus activating them and initiating those adjacent nodes if necessary. The agent also performs the consensus iteration (8) over its active links in each time step $T > t$. In this manner, the INITIATE message 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 Appendix C. One can think of diffusive distributed consensus as a standard consensus algorithm over a graph $(V, \overline{E}^{(t)})$, where $\overline{E}^{(t)} \subseteq E^{(t)}$ contains only active links.

In a time-varying network, an agent may not receive the INITIATE message containing $x_1^{(k)}$ in every iteration. Furthermore, it may receive an INITIATE message for $x_1^{(j)}$, with $j < k$, after it receives $x_1^{(k)}$. The iteration number k is included in each INITIATE message so that an agent can determine whether the message contains the most up-to-date iterate it has seen so far. If it does, then the agent uses this iterate to compute its intermediate vector and begins the consensus algorithm with its active neighbors. Otherwise, the agent ignores the message.

To ensure convergence of the diffusive distributed consensus algorithm, we require that the time-varying network satisfy the network connectivity conditions in Assumption 2. A consequence of this assumption is that, after at most $C(P-1)$ time steps, every agent is activated. Therefore, for $\overline{D} = 2(P-1)C$, we have $\overline{E}^{(t)} \cup \overline{E}^{(t+1)} \cup \cdots \cup \overline{E}^{(t+\overline{D}-1)} = E^{(\infty)}$.

We also make the following standard assumption on the weights used in iteration (8) [14], [23].

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

(a) The matrix $W^{(t)}$ is doubly stochastic.

(b) There exists a scalar $\eta \in (0,1)$, such that for all p, $w_{pp}^{(t)} \geq \eta$. Further, if $(p,q) \in \overline{E}^{(t)}$, then $w_{pq}^{(t)} \geq \eta$, and if $(p,q) \notin \overline{E}^{(t)}$, then $w_{pq}^{(t)} = 0$.

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 time steps of diffusive distributed consensus.

Proposition 4.3: Let the network satisfy Assumption 2. After s time steps 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,

$$\|v_p^{(s)} - v_{av}\|_2 \leq \Gamma \gamma^s \sum_{q=1}^{P} \|v_q^{(0)}\|_2, \quad \text{for } p = 1, \ldots P,$$

where $\Gamma = 2(1+\eta^{-\overline{D}})/(1-\eta^{\overline{D}})$ and $\gamma = (1-\eta^{\overline{D}})^{1/\overline{D}}$, with $\overline{D} = 2(P-1)C$.

This proposition is a straightforward extension of Proposition 1 in [14] for the standard distributed consensus algorithm. We therefore omit the proof for brevity.

C. Consensus-Based DIHT

We now detail our CB-DIHT algorithm. As in DIHT for static networks, each agent has an estimate $x_p^{(k)}$, initially x_{init}. For each iteration k of CB-DIHT, agent 1 computes its intermediate vector according to (5). It 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 updates its local estimate to be identical to that of agent 1, i.e., it sets $x_p^{(k)} = x_1^{(k)}$. 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,

$$x_1^{(k+1)} = \mathcal{T}_K \left(x_1^{(k)} - \tfrac{1}{L_{TV}} \hat{v}^{(k)} \right), \tag{9}$$

Algorithm 2: Consensus-Based DIHT.

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

Algorithm executed by agent 1.
while TRUE do
    $z_1^{(k)} \leftarrow \nabla f_1(x_1^{(k)})$                    Local computation.
    $s^{(k)} \leftarrow \left\lceil \frac{1}{2}(k + \|x_1^{(k)}\|_2^2) \right\rceil$    Number of consensus steps.
    $k_1 \leftarrow k$
    $\hat{v}^{(k)} \leftarrow$ DiffusDistConsensus$(k_1, x_1^{(k)}, z_1^{(k)}, s^{(k)})$
    $x_1^{(k+1)} \leftarrow \mathcal{T}_K\left(x_1^{(k)} - \frac{1}{L_{TV}}\hat{v}^{(k)}\right)$
    $k \leftarrow k+1$

Algorithm executed by agent $p \neq 1$.
while TRUE do
    on receive$_p(k_1, x_1^{(k)})$
        if $k_1 > k$ then
            stop DiffusDistConsensus for iter. $k$
            $k \leftarrow k_1$
            $x_p^{(k)} \leftarrow x_1^{(k_1)}$
            $z_p^{(k)} \leftarrow \nabla f_p(x_p^{(k)})$        Local computation.
            In next time step,
              DiffusDistConsensus $(k_1, x_1^{(k)}, z_p^{(k)})$
```

where $L_{TV} > \frac{1}{P}L_f$. It then begins a new instance of diffusive distributed consensus for iteration $k+1$ of CB-DIHT. Pseudocode for CB-DIHT is given in Algorithm 2.

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 $\hat{v}^{(k)}$.

For each iteration k, the estimates at agents $p \neq 1$ are identical to those at agent 1. We note that, due to the time-varying nature of the network, it is possible that some agents may not be initiated in the distributed diffusive consensus instance for a given iteration. Therefore, the estimates at these agents may skip iterations of IHT. By Assumption 2, each agent's estimate will be updated in infinitely many iterations, and so it suffices to analyze the convergence of the estimate at agent 1. This estimate evolves as follows,

$$s^{(k)} = \left\lceil \tfrac{1}{2}(k + \|x_1^{(k)}\|_2^2) \right\rceil \tag{10}$$

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

$$x_1^{(k+1)} = \mathcal{T}_K\left(x_1^{(k)} - \tfrac{1}{L_{TV}}\hat{v}^{(k)}\right), \tag{12}$$

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

$$\Phi(s^{(k)}) = W^{(t_{s^{(k)}})}\, W^{(t_{s^{(k)}-1})} \cdots W^{(t_2)}\, W^{(t_1)},$$

with each $W^{(t_i)}$ satisfying Assumption 3. The notation $[\,\cdot\,]_{1p}$ indicates the entry of the 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

$$x_1^{(k+1)} = \mathcal{T}_K\left(x_1^{(k)} - \tfrac{1}{L}\left(\nabla f(x_1^{(k)}) + \epsilon^{(k)}\right)\right), \tag{13}$$

where $L = PL_{TV}$, $f(x_1^{(k)}) = \sum_{p=1}^{P} f_p(x_1^{(k)})$, and $\epsilon^{(k)} = P\hat{v}^{(k)} - \sum_{p=1}^{P}\nabla f_p(x_1^{(k)})$.

Proof: By (12), the vector $x_1^{(k)}$ evolves as,

$$\begin{aligned} x_1^{(k+1)} &= \mathcal{T}_K\left(x_1^{(k)} - \tfrac{1}{L_{TV}}\hat{v}^{(k)}\right) \\ &= \mathcal{T}_K\left(x_1^{(k)} - \tfrac{1}{L_{TV}}\left(\tfrac{1}{P}\nabla f(x_1^{(k)}) + \hat{v}^{(k)} - \tfrac{1}{P}\nabla f(x_1^{(k)})\right)\right) \\ &= \mathcal{T}_K\left(x_1^{(k)} - \tfrac{1}{PL_{TV}}\left(\nabla f(x_1^{(k)}) + P\hat{v}^{(k)} - \nabla f(x_1^{(k)})\right)\right). \end{aligned}$$

Substituting with the expressions, $L = PL_{TV}$ and $\epsilon^{(k)} = P\hat{v}^{(k)} - \sum_{p=1}^{P}\nabla f_p(x_1^{(k)})$, we obtain (13). ∎

We now show that, under Assumptions 1, 2, and 3, 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 $\sum_{k=0}^{\infty}\|\epsilon^{(k)}\|_2^2 < \infty$.

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

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

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

$$\epsilon^{(k)} = P\left(\hat{v}^{(k)} - \overline{v}^{(k)}\right).$$

Using Proposition 4.3, we bound $\|\epsilon^{(k)}\|_2$ as

$$\|P(\hat{v}^{(k)} - \overline{v}^{(k)})\|_2 \leq P\Gamma\gamma^{s^{(k)}}\sum_{p=1}^{P}\|\nabla f_p(x_1^{(k)})\|_2.$$

Therefore,

$$\sum_{k=0}^{\infty}\|P(\hat{v}^{(k)} - \overline{v}^{(k)})\|_2^2 \leq \sum_{k=0}^{\infty}\left(P\Gamma\gamma^{s^{(k)}}\sum_{p=1}^{P}\|\nabla f_p(x_1^{(k)})\|_2\right)^2 \tag{14}$$

$$\leq P^2\Gamma^2\sum_{k=0}^{\infty}\gamma^{2s^{(k)}}(G\|x_1^{(k)}\|_2 + H)^2, \tag{15}$$

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

Substituting the value of $s^{(k)}$ from (10) into (15), we obtain

$$\begin{aligned}\sum_{k=0}^{\infty}\|P(\hat{v}^{(k)}-\overline{v}^{(k)})\|_2^2 \leq\ & P^2\Gamma^2G^2\sum_{k=0}^{\infty}\gamma^k\|x_1^{(k)}\|_2^2\gamma^{\|x_1^{(k)}\|_2^2} \\ &+2P^2\Gamma^2GH\sum_{k=0}^{\infty}\gamma^k\|x_1^{(k)}\|_2\gamma^{\|x_1^{(k)}\|_2^2} \\ &+P^2\Gamma^2H^2\sum_{k=0}^{\infty}\gamma^k\gamma^{\|x_1^{(k)}\|_2^2}.\end{aligned}$$

We note that since $\gamma \in (0,1)$, the functions $y\gamma^y$, $\sqrt{y}\gamma^y$, and γ^y are bounded for any $y \geq 0$ (including $y = \|x_1^{(k)}\|_2^2$). Thus, the sum of the sequence $\{\|\hat{v}^{(k)}-\overline{v}^{(k)}\|_2^2\}_{k\geq 0}$ is upper-bounded by the sum of a geometric sequence $\{J\gamma^k\}_{k\geq 0}$ for some constant J. Since $\gamma \in (0,1)$, this sequence is summable, and thus $\{\|P(\hat{v}^{(k)}-\overline{v}^{(k)})\|_2^2\}_{k\geq 0}$ is 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-DIHT with $L_{TV} > \frac{1}{P}L_f$. Then,

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), with spark$(A) > K$, the sequences $\{x_p^{(k)}\}_{k\geq 0}$, $p = 1\dots P$, converge to an L-stationary point.

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

V. Simulations

In this section, we demonstrate the performance of our distributed algorithms for several recovery problems. We also compare our algorithms with previously proposed distributed approaches 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 method that addresses this general problem. Therefore we restrict our evaluations to distributed compressed sensing, for which there are several other existing algorithms. We now briefly review these other methods.

A. Alternative Algorithms

As discussed in Section I, other algorithms for distributed compressed sensing use a convex optimization formulation, for example, basis pursuit [28]:

$$\begin{aligned}&\text{minimize } \tfrac{1}{P}\textstyle\sum_{p=1}^{P}\|x\|_1 \\ &\text{subject to } A_px = b_p, \quad p = 1\dots P.\end{aligned} \tag{16}$$

It has been shown that basis pursuit algorithms have recovery guarantees comparable to centralized IHT [1]. Therefore, distributed basis pursuit and DIHT also have comparable recovery guarantees. CB-DIHT is based on a generalization of centralized IHT for nonlinear objectives, and the recovery guarantees of this version of IHT are not as well studied. In our evaluations, DIHT and CB-DIHT exhibited similar signal recovery capabilities.

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

In a recent work, Mota et al. compared several distributed basis pursuit algorithms for static networks and 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 in our evaluation, and we repeat the same experiments here.

For time-varying networks, we compare CB-DIHT with the distributed subgradient algorithm [15]. This algorithm was proposed to solve a general class of convex optimization problems, of which, (16) is a special case. Previous work has shown that, in time-varying networks, the subgradient algorithm outperforms the double-looped method in [13] in similar evaluations.

We briefly describe each of these algorithms. Pseudocode is Appendix D.

1) D-ADMM: 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, $x_p^{(k)}$. In every iteration, the agent exchanges an intermediate N-vector with all of its neighbors, according to the order dictated by the graph coloring, and it generates its next iterate by solving a local convex optimization problem. In a single iteration of D-ADMM, only one color of agents sends messages at a time. Therefore, one iteration of D-ADMM takes c times as long as one iteration of distributed consensus. We note that, it has not been theoretically verified that D-ADMM converges to the optimal solution of (16) in general graphs, however its convergence has been demonstrated experimentally [7].

2) Distributed Subgradient Algorithm: In the distributed subgradient algorithm, each agent p has an estimate $x_p^{(k)}$, initially 0. Every agent performs a single step of the distributed consensus algorithm to form a weighted average of its and its neighbors' estimates. Thus, in every iteration, the agent receives an N-vector along all of its incoming links in that iteration. The agent then locally computes $x_p^{(k+1)}$ by performing a projected gradient step with step size $\alpha^{(k)}$. If the weights used for the consensus algorithm satisfy Assumption 3 and the step-size sequence $\{\alpha^{(k)}\}_{k\geq 0}$ is square-summable but not summable, then, in a static network, this algorithm converges to the optimal solution of (16). Convergence has also been shown in time-varying networks that obey Assumption 2 [15].

B. Evaluation Setup

We show evaluation results for three compressed sensing problems from the Sparco toolbox [29]. Details of the problems are given in Table I. For each problem, we use the measurement matrix A and original, sparse signal x provided by the toolbox. We generate the measurement vector $b = Ax$ without any measurement noise. For each problem, we divide the measurements (rows from A and b) 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 class, we generate five random instances. The results shown in this section are the averages of the five runs over the five graph instances. The first graph type is a Barabasi-Albert (BA) scale free graph [30]. The second and third are Erdös-Rényi (ER) random graphs [31] 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 [32] with vertices placed uniformly at random in a unit square. In the fourth graph, two vertices are connected if they are within a distance $d = 0.5$ of each other, and in the fifth, vertices are connected if they are within a distance $d = 0.75$. 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, with 992 edges, on average, for $N = 50$ and 1,514 edges, on average, for $N = 64$.

For 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, one per time step.

We have implemented all algorithms in Matlab and use CVX [33] to solve the local optimization problems in D-ADMM. All algorithms are initiated with each agent's estimate equal to 0. D-ADMM requires a graph coloring, which we generate using the heuristic from the Matgraph toolbox [34], as 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, we set $L = 2.01$ for Sparco problems 902 and 7. For Sparco problem 11, we use two values of L, $L = 4570$ and $L = 500$. For CB-DIHT in static networks, we let $L_{TV} = L/P$, where L is as for DIHT. For CB-DIHT in time-varying networks, we use $L_{TV} = 2.01/P$ for Sparco problems 902 and 7. For Sparco problem 11, we use $L_{TV} = 4570/P$ and $L_{TV} = 600/P$. For problem 11, the smaller values of L and L_{TV} are not sufficient to guarantee convergence. In our evaluations, for these values, DIHT and CB-DIHT converge to the original signal. For both DIHT and CB-DIHT, K is set to the value in Table I.

For the distributed subgradient algorithm, we experimented with different step-sizes $\alpha^{(k)} = \frac{1}{k^a}$, 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 was with $a = 0.6$. In the results below, we use $a = 0.7$, which was the value with the second fastest convergence for the vast majority of graphs. For D-ADMM, we set the algorithm parameters to those that were shown to be best in the same experiments [7].

C. Results for Static Networks

For each algorithm, we measure the number of values sent for $\|x_p^{(t)} - x^*\|/\|x^*\|$ to be less than either 10^{-2} or 10^{-5} at every agent. For D-ADMM and the subgradient algorithm, x^* is the original sparse signal from the Sparco toolbox[1]. 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 x^* where applicable. In most experiments, DIHT and CB-DIHT do converge to the original sparse signal. 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 DIHTand CB-DIHT, some messages consist of K values and others consists of N values, while in 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. When an agent sends a K-sparse vector, this requires $2K$ messages; each component of the vector requires two messages, one containing the index in the vector and one containing the corresponding value. Results on the total number of messages that would be sent using a broadcast message model are given in Appendix E. We also measure the time to convergence in a synchronous network where each message is delivered in one time step. For all algorithms, we allow one value to be sent on a given link in each direction per time step.

We compute the number of of values transmitted by each algorithm as follows. For DIHT, each iteration consists of a broadcast phase and convergecast phase. In the broadcast phase, each agent p sends a K-vector to all of its children, requiring $2KQ_p$ messages where Q_p is the number of children that agent p has in the spanning tree (note that Q_p is less than or equal to the node degree of agent p in the original graph). In the convergecast phase, each agent, except agent 1, sends an N-vector to its parent in the tree, requiring N messages. In a network of P agents, a spanning tree has $P - 1$ edges. Therefore, for a given problem, DIHT requires the same number of messages for convergence for every network topology. The number of messages needed to create the spanning tree depends on the network topology, but this messages count is insignificant when compared message count of the algorithm execution. In D-ADMM and the subgradient algorithm, each agent sends an N-vector to all of its neighbors in the original graph in each iteration. Thus, each agent sends $\Delta_p N$ messages per iteration, where Δ_p is the node degree of agent p in the original graph. In CB-DIHT, each agent p sends at most $\Delta_p - 1$ activation vectors per iteration, where each activation vector is K-vector that is sent in $2K$ messages. After activation, an agent sends N-vectors to each of its active neighbors in each distributed diffusive consensus round, sending at most $\Delta_p N$ messages per consensus round.

The results for Sparco problems 902 and 7 are shown in Tables II and III. For both problems, DIHT outperforms all other algorithms in both bandwidth and time on all graph instances,

[1]We have numerically verified that a centralized basis pursuit formulation recovers this original sparse signal. Additionally, it has been shown in [7] that, in the same simulation setup, D-ADMM and the distributed subgradient method converge to the original signal for Sparco problems 902, 7, and 11.

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 for each algorithm to converge to within the specified accuracy.

	Accuracy 10^{-2}				**Accuracy** 10^{-5}			
Graph	**DIHT**	**D-ADMM**	**CB-DIHT**	**Subgrad.**	**DIHT**	**D-ADMM**	**CB-DIHT**	**Subgrad.**
BA	2.32×10^6	2.01×10^7	2.27×10^8	$>5 \times 10^{10}$	5.82×10^6	2.31×10^7	7.78×10^8	$>5 \times 10^{10}$
ER (*pr*=0.25)	2.32×10^6	5.30×10^7	8.45×10^8	2.40×10^{10}	5.82×10^6	7.31×10^7	3.18×10^9	$>1 \times 10^{11}$
ER (*pr*=0.75)	2.32×10^6	3.26×10^8	1.34×10^9	9.49×10^9	5.82×10^6	3.85×10^8	6.75×10^9	$>3 \times 10^{11}$
Geo (*d*=0.5)	2.32×10^6	3.29×10^7	1.65×10^9	$\geq 5.3 \times 10^9$	5.82×10^6	3.80×10^7	3.74×10^9	$>7 \times 10^{10}$
Geo (*d*=0.75)	2.32×10^6	1.80×10^8	1.38×10^9	1.11×10^{10}	5.82×10^6	2.19×10^8	5.72×10^9	$>4 \times 10^{11}$

(b) Total number of time steps for each algorithm to converge to within the specified accuracy.

	Accuracy 10^{-2}				**Accuracy** 10^{-5}			
Graph	**DIHT**	**D-ADMM**	**CB-DIHT**	**Subgrad.**	**DIHT**	**D-ADMM**	**CB-DIHT**	**Subgrad.**
BA	1.51×10^5	3.14×10^5	1.02×10^5	$>4 \times 10^8$	3.80×10^5	3.60×10^5	4.39×10^6	$>4 \times 10^8$
ER (*pr*=0.25)	1.23×10^5	7.55×10^5	1.58×10^6	7.94×10^7	3.09×10^5	8.63×10^5	5.61×10^6	$>4 \times 10^8$
ER (*pr*=0.75)	9.46×10^4	3.14×10^6	8.36×10^5	2.91×10^7	2.37×10^5	3.71×10^6	3.91×10^6	$>4 \times 10^8$
Geo (*d*=0.5)	2.22×10^5	6.92×10^5	4.95×10^6	$\geq 2.8 \times 10^7$	8.07×10^5	7.98×10^5	1.09×10^7	$>4 \times 10^8$
Geo (*d*=0.75)	9.46×10^4	2.75×10^6	1.21×10^6	3.40×10^7	2.37×10^5	3.36×10^6	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 for each algorithm to converge to within specified accuracy.

	Accuracy 10^{-2}				**Accuracy** 10^{-5}			
Graph	**DIHT**	**D-ADMM**	**CB-DIHT**	**Subgrad.**	**DIHT**	**D-ADMM**	**CB-DIHT**	**Subgrad.**
BA	6.12×10^6	7.01×10^7	1.57×10^9	$>1 \times 10^{11}$	1.64×10^7	1.17×10^8	5.23×10^9	$>1 \times 10^{11}$
ER (*pr*=0.25)	6.12×10^6	2.52×10^8	3.76×10^9	$>3 \times 10^{11}$	1.64×10^7	3.07×10^8	1.24×10^{10}	$>3 \times 10^{11}$
ER (*pr*=0.75)	6.12×10^6	9.28×10^8	1.16×10^{10}	3.63×10^{10}	1.64×10^7	1.78×10^9	3.80×10^{10}	$>9 \times 10^{11}$
Geo (*d*=0.5)	6.12×10^6	1.23×10^8	7.45×10^9	$>1 \times 10^{11}$	1.64×10^7	1.61×10^8	5.85×10^{10}	$>1 \times 10^{11}$
Geo (*d*=0.75)	6.12×10^6	4.53×10^8	7.99×10^9	6.25×10^{10}	1.64×10^7	7.51×10^8	2.65×10^{10}	$>1 \times 10^{12}$

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

	Accuracy 10^{-2}				**Accuracy** 10^{-5}			
Graph	**DIHT**	**D-ADMM**	**CB-DIHT**	**Subgrad.**	**DIHT**	**D-ADMM**	**CB-DIHT**	**Subgrad.**
BA	3.99×10^5	1.32×10^6	6.62×10^6	$>1 \times 10^9$	1.07×10^6	1.72×10^6	2.13×10^7	$>1 \times 10^9$
ER (*pr*=0.25)	3.25×10^5	2.98×10^6	6.73×10^6	$>1 \times 10^9$	8.72×10^5	3.63×10^6	2.14×10^7	$>1 \times 10^9$
ER (*pr*=0.75)	2.50×10^5	8.97×10^6	6.74×10^6	3.93×10^7	6.71×10^5	1.64×10^7	2.15×10^7	$>1 \times 10^9$
Geo (*d*=0.5)	8.49×10^5	2.98×10^6	2.16×10^7	$>1 \times 10^9$	2.28×10^6	3.38×10^6	4.84×10^7	$>1 \times 10^9$
Geo (*d*=0.75)	2.50×10^5	1.46×10^7	6.62×10^6	9.82×10^7	6.71×10^5	2.32×10^7	2.13×10^7	$>1 \times 10^9$

and in all cases, DIHT recovers the optimal solution. DIHT requires two orders of magnitude fewer values and 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 time steps than D-ADMM to achieve an accuracy of 10^{-5}. Both CB-DIHT and the subgradient algorithm require more values and time 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 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 required by this algorithm. 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 in most cases.

Results for Sparco problem 11 are shown in Table IV. For DIHT, $L = 4570$ is sufficient to guarantee convergence to an L-stationary point. However, convergence with this L is slower than that of D-ADMM, sometimes requiring up to one order of magnitude more time steps, although with less bandwidth. Also, with $L = 4570$, DIHT converged to an L-stationary point that was suboptimal. In all simulations, DIHT with $L = 500$ converged to the original signal. In addition, with the smaller L, DIHT sent one to three orders of magnitude fewer values than D-ADMM and required one to three orders of magnitude fewer time steps. The performance of CB-DIHT is also significantly worse for $L_{TV} = 4570/P$ than for $L_{TV} = 500/P$, by several orders of magnitude. Additionally, for $L_{TV} = 4570/P$, CB-DIHT converged to a suboptimal L-stationary point in all but two graph instances. With $L_{TV} = 500/P$, CB-DIHT converged to the original

TABLE IV: Signal recovery in a static network for Sparco problem 11 to an accuracy of 10^{-2}. For DIHT with $L = 4750$ and CB-DIHT with $L_{TV} = 4570/P$, in the vast majority of experiments, the algorithms converge to an L-stationary point that is not the original signal. The values shown for DIHT and CB-DIHT are for convergence to the L-stationary point; these values are preceded by a †. For convergence to the original signal, the values in these columns would all be infinite. For all other columns, the values shown are for convergence to the original signal.

(a) Total number of values transmitted for each algorithm to converge.

Graph	**DIHT** $L = 4570$	**DIHT** $L = 500$	**D-ADMM**	**CB-DIHT** $L_{TV} = 4570/P$	**CB-DIHT** $L_{TV} = 500/P$	**Subgradient**
BA	$^{\dagger}3.37 \times 10^{7}$	1.58×10^{6}	8.15×10^{7}	$^{\dagger}2.27 \times 10^{10}$	3.06×10^{8}	$>5 \times 10^{10}$
ER (*pr*=0.25)	$^{\dagger}3.37 \times 10^{7}$	1.58×10^{6}	5.30×10^{8}	$^{\dagger}6.60 \times 10^{10}$	6.96×10^{8}	$>1 \times 10^{11}$
ER (*pr*=0.75)	$^{\dagger}3.37 \times 10^{7}$	1.58×10^{6}	4.83×10^{9}	$^{\dagger}4.95 \times 10^{10}$	3.66×10^{9}	1.56×10^{11}
Geo (*d*=0.5)	$^{\dagger}3.37 \times 10^{7}$	1.58×10^{6}	2.26×10^{8}	$^{\dagger}4.34 \times 10^{10}$	2.54×10^{10}	$> 7 \times 10^{10}$
Geo (*d*=0.75)	$^{\dagger}3.37 \times 10^{7}$	1.58×10^{6}	2.91×10^{9}	$^{\dagger}7.73 \times 10^{10}$	1.50×10^{9}	4.02×10^{10}

(b) Total number of time steps for each algorithm to converge.

Graph	**DIHT** $L = 4570$	**DIHT** $L = 500$	**D-ADMM**	**CB-DIHT** $L_{TV} = 4570/P$	**CB-DIHT** $L_{TV} = 500/P$	**Subgradient**
BA	$^{\dagger}1.61 \times 10^{6}$	7.53×10^{4}	9.49×10^{5}	$^{\dagger}6.76 \times 10^{7}$	1.27×10^{6}	$>4 \times 10^{8}$
ER (*pr*=0.25)	$^{\dagger}1.39 \times 10^{6}$	6.52×10^{4}	4.42×10^{6}	$^{\dagger}6.75 \times 10^{7}$	1.22×10^{6}	$>4 \times 10^{8}$
ER (*pr*=0.75)	$^{\dagger}1.07 \times 10^{6}$	5.02×10^{4}	3.56×10^{7}	$^{\dagger}1.68 \times 10^{7}$	1.16×10^{6}	1.04×10^{8}
Geo (*d*=0.5)	$^{\dagger}3.11 \times 10^{6}$	1.46×10^{5}	3.57×10^{6}	$^{\dagger}7.51 \times 10^{7}$	6.96×10^{7}	$>4 \times 10^{8}$
Geo (*d*=0.75)	$^{\dagger}1.28 \times 10^{6}$	6.02×10^{4}	3.29×10^{7}	$^{\dagger}3.66 \times 10^{7}$	1.23×10^{6}	3.48×10^{7}

signal in all cases. These results indicate that the bound on L in Theorem 4.1 is not tight, and that further investigation into the convergence conditions for both IHT and its distributed variants is warranted. For the BA graph, the ER graph with $pr = 0.25$ and the geometric graph with $d = 0.5$, the subgradient algorithm required more than $2 \cdot 10^{5}$ iterations to converge to within an error of 10^{-2}. As the table shows, CB-DIHT with the larger L_{TV} outperformed the subgradient method in both time and bandwidth for all but one graph (Geo $d = 0.75$). With the smaller value of L_{TV}, CB-DIHT outperformed the subgradient method in both time and bandwidth, usually by at least one order of magnitude.

One interesting observation is that for DIHT, CB-DIHT, and the distributed subgradient algorithm, as the network connectivity increases, both the bandwidth and convergence time 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, a more connected the network results in a spanning tree with a smaller height; thus, less bandwidth and time is needed to compute each sum. CB-DIHT and the subgradient algorithm both employ distributed consensus algorithms which are well known to 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.

In D-ADMM, the agents' estimates should become sparse as the algorithm progresses. If this happens, the agents can exchange sparse representations of their estimates, thus reducing the total number of messages. We checked for this in our simulations, and for the problems and networks considered herein, using sparse representations had little impact on the evaluation results. We believe that it is possible to optimize the communication of D-ADMM in some settings using sparse vector representations, however, this optimization is beyond the scope of this paper.

Finally, we note that, for some network messaging schemes, such as TDMA, it is more efficient to send messages containing multiple values, rather than a single value per message. All algorithms studied in this section would see improvement under this type of scheme. Since in DIHT, a K-sparse vector is broadcast in each iteration, while the other algorithms always send N-vectors, DIHT would not benefit as much.

D. Results for Time-Varying Networks

We compare 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. For each algorithm, we count the number of consensus rounds needed for $\|x_p^{(t)} - x^*\|/\|x^*\|$ to be less than either 10^{-2} or 10^{-5} at every agent, where 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 $3 \cdot 10^{5}$ consensus rounds. For the subgradient algorithm, every graph instance required more than $3 \cdot 10^{5}$ consensus rounds to converge to within 10^{-5} of x^*. Therefore, we do not show these values in the table. For Sparco problems 902 and 7, CB-DIHT converged

TABLE V: Number of iterations of distributed consensus needed for signal recovery in a time-varying network. For 10^{-5} accuracy, the subgradient algorithm required more than $3 \cdot 10^5$ iterations in every instance. For Sparco problem 11 only, CB-DIHT with $L_{TV} = 4570/P$ does not always converge to the original signal. The values shown in this column are for convergence to a sub-optimal L-stationary point. These values are preceded by a †. All other columns give values for convergence to the original signal.

(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	6.8×10^4	1.0×10^4

(c) Sparco problem 11

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

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 before, for CB-DIHT on Sparco problem 11, we use a value of L_{TV} that is sufficient to guarantee convergence, $L_{TV} = 4750/P$, and a smaller value, $L_{TV} = 600/P$, that is not sufficient to guarantee convergence, but nevertheless, converges in all experiments. For the larger value of L_{TV}, CB-DIHT converged to a suboptimal L-stationary point in all but four experiments. For $L_{TV} = 600/P$, CB-DIHT always converged to the original signal. For both values of L_{TV}, CB-DIHT required fewer consensus rounds to converge than the subgradient algorithm. This difference is more pronounced with $L_{TV} = 600/P$, 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 into the relationship between L_{TV} and the convergence behavior of CB-DIHT.

VI. Discussion of Fault Tolerance

In both DIHT and CB-DIHT, agent 1 is solely responsible for performing the thresholding operation in each iteration. A natural question that arises is what happens if this agent fails, or more generally, are these algorithms fault tolerant?

In a discussion of fault tolerance in a static network, we must first assume that the network is synchronous since it is impossible to detect node failures in an asynchronous network. Under this assumption, it is straightforward to make DIHT fault tolerant using a self-stabilizing, distributed algorithm for constructing the spanning tree [35]. The network will autonomously reconstruct the tree on detection of an agent failure, so long as the underly graph remains connected. Should the root fail (agent 1), the new root will assume the role of agent 1. Since the agents all share the same estimate, once the tree is repaired, the algorithm can pick up essentially where it left off. While the tree is under repair, some agents' estimates may diverge. However, since there is a single leader and every agent has a single parent in the tree, this hierarchy ensures that the system will return to a consistent state.

After a failure, the objective function will be different; the current estimate serves as warm start for the new optimization problem. In addition, if agent 1 uses $\sum_{p=1} L_{f_p}$ as its upper bound for L_f in the original problem, this sum is also an upper bound for L_f for the problem after the failure. Therefore, the value of L does not need to change.

In time-varying networks, it is not possible to detect failures. Therefore, there is no straightforward way to make CB-DIHT handle the failure of agent 1. If any other agent fails, CB-DIHT can proceed without modification, using the current estimate as the initial estimate for the new optimization problem.

VII. 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. Next, we proposed an extension of DIHT for time-varying networks. We 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 algorithms in both bandwidth and time by several orders of magnitude. In time-varying networks, CB-DIHT outperforms the best known algorithm for distributed compressed sensing that accommodates changing network topologies. We 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.

Appendix A
Distributed Computation of L and L_{TV}

To determine L or L_f, agent 1 must learn an upper bound on the Lipschitz constant L_f of the function $f(x) = \sum_{p=1}^P f_p(x)$. We first note that, by Assumption 1(b), for all $x, y \in \mathbb{R}^N$,

$$\begin{aligned}\left\|\sum_{p=1}^P \nabla f_p(x) - \sum_{p=1}^P \nabla f_p(y)\right\| &\leq \sum_{p=1}^P \|\nabla f_p(x) - \nabla f_p(y)\| \\ &\leq \sum_{p=1}^P L_{f_p}\|x-y\|.\end{aligned}$$

Therefore, $\sum_{p=1}^P L_{f_p}$ is an upper bound for L_f. We now present distributed algorithms by which agent 1 can learn this sum.

Computation for DIHT: As a pre-processing step for DIHT, the agents construct a spanning tree of the graph G with agent 1 as its root. For agent 1 to learn the sum $\sum_{p=1}^P L_{f_p}$, it simply needs to broadcast a request down the tree. The agents then use a convergecast to aggregate their values for L_{f_p} up the tree. The aggregation convergecast is identical to that used to compute the sum of intermediate vectors $\sum_{p=1}^P z_p^{(k)}$ in each iteration of DIHT, as described in Section III-B. Once agent 1 knows this sum, it can select an $L > \sum_{p=1}^P L_{f_p}$ so that the convergence of DIHT is guaranteed.

Computation for CB-DIHT: For CB-DIHT, the step size L_{TV} must be such that $L_{TV} > \frac{1}{P} L_f$. One possibility is for the agents to use a distributed consensus algorithm to estimate the average of their respective L_{f_p} However, agent 1 may not be able to determine how many consensus rounds are needed to estimate the average with enough accuracy to generate a correct upper bound. A more communication efficient option is for agent 1 to use a distributed algorithm to find $L_{max} = \max\{L_{f_1}, \ldots, L_{f_P}\}$. Since $L_{max} \geq \frac{1}{P}\sum_{p=1}^P L_{f_p}$, L_{max} can be used as an (non-strict) upper bound on the average.

To compute L_{max}, every agent stores a variable m_p that it initializes to L_{f_p}. In every round t, the agents sends m_p to all neighbors in that time step. When an agent receives a value m_q from a neighbor, where $m_q > m_p$, it sets $m_p = m_q$. If the network satisfies Assumption 2, then after at most $2C\Delta$ rounds of the algorithm, where C is as defined in Assumption 2 and Δ is the diameter of the graph $(V, E^{(\infty)})$, $m_1 = L_{max}$, i.e., agent 1 knows the correct value for L_{max}.

Since the agents do not know C or Δ, agent 1 cannot determine how long to wait before it learns L_{max} and can begin the CB-DIHT algorithm. Instead, the agents execute the distributed max-finding algorithm concurrently with CB-DIHT. In each iteration k of IHT, agent 1 uses its current value m_1 as a bound for the step size, i.e., $L_{TV} > m_1$. After a finite number of time steps, $m_1 = L_{max}$. Therefore, the convergence guarantees stated in Theorems 4.1 and 4.2 still hold.

Appendix B
Proofs for Iterative Hard Thresholding with Inexact Gradients

In this section, $\|\cdot\|$ denotes the ℓ_2 norm.

First, for convenience, we restate some relevant results from [36] and [10].

Lemma B.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 $L \geq L_f$,

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

where

$$h_L(x,y) := f(y) + (x-y)^\mathsf{T}\nabla f(y) + \tfrac{L}{2}\|x-y\|^2. \qquad (17)$$

Lemma B.2 (Lemma 2.2 from [10]): For any $L > 0$, x^* is an L-stationary point of problem (1) if and only if $\|x^*\|_0 \leq K$ and, for $i = 1 \ldots N$,

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

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 2.4 in [10]).

Lemma B.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 $k \geq 0$:

$$\begin{aligned} f(x^{(k)}) - f(x^{(k+1)}) \geq \tfrac{L-L_f}{2}\|x^{(k)} - x^{(k+1)}\|^2 \\ - \left(x^{(k)} - x^{(k+1)}\right)^\mathsf{T} \epsilon^{(k)}. \quad (18)\end{aligned}$$

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

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

where h_L is as defined in (17). The above implies that,

$$h_L(x^{(k+1)}, x^{(k)}) + (x^{(k+1)} - x^{(k)})^\intercal \epsilon^{(k)} \le h_L(x^{(k)}, x^{(k)}) = f(x^{(k)}). \quad (19)$$

By Lemma B.1, we have

$$\begin{aligned} f(x^{(k)}) - f(x^{(k+1)}) &\ge f(x^{(k)}) - h_{L_f}(x^{(k+1)}, x^{(k)}) \quad (20)\\ &\ge f(x^{(k)}) - h_L(x^{(k+1)}, x^{(k)}) \\ &\quad + \tfrac{L-L_f}{2}\|x^{(k+1)} - x^{(k)}\|^2, \quad (21)\end{aligned}$$

where (21) is obtained from (20) by applying the identity,

$$h_{L_f}(x,y) = h_L(x,y) - \tfrac{L-L_f}{2}\|x-y\|^2.$$

Combining (19) and (21), we obtain the result in (18). ∎

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

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

1) There exists a $D < \infty$ such that for all $T \ge 0$, $\sum_{k=0}^{T}\|x^{(k)} - x^{(k+1)}\|^2 \le D$.
2) $\lim_{k\to\infty}\|x^{(k)} - x^{(k+1)}\|^2 = 0$.

Proof:

To prove the first part of the lemma, we show the sequence $\{\sum_{k=0}^{T}\|x^{(k)} - x^{(k+1)}\|^2\}_{T\ge 0}$ is bounded. Consider the sum over time of $f(x^{(k)}) - f(x^{(k+1)})$. We can bound this as,

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

where the last inequality follows from the fact that f is lower bounded by a constant B. This bound holds for all $T \ge 0$.

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

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

$$\ge \tfrac{L-L_f}{2}\sum_{k=0}^{T}\|x^{(k)} - x^{(k+1)}\|^2 - \sqrt{E\sum_{k=0}^{T}\|x^{(k)} - x^{(k+1)}\|^2}, \quad (23)$$

where (23) follows from (22) by the Cauchy-Schwarz inequality and 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 (23) as,

$$-\beta D^2 + CD + A \ge 0. \quad (24)$$

Our goal is to show that the sum D is bounded (for all $T \ge 0$), i.e., we must show that every $D \ge 0$ that satisfies (24) is bounded. These values are such that,

$$0 \le D \le \left(C + \sqrt{C^2 + 4\beta A}\right)/(2\beta).$$

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

We now show that $\{\sum_{t=0}^{T}\|x^{(k)} - x^{(k+1)}\|^2\}_{T\ge 0}$ converges, implying part two of the lemma (see [37], Theorem 3.23). To this end, we must show that the sequence is monotonically non-decreasing and bounded. We have already established that it is bounded. Monotonicity is easily established by,

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

∎

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 $\{x^{(k)}\}_{k\ge 0}$ be the sequence generated by (7) with $L > L_f$ and with a sequence $\{\epsilon^{(k)}\}_{k\ge 0}$ satisfying $\sum_{k=0}^{\infty}\|\epsilon^{(k)}\|_2^2 < \infty$. Then, any accumulation point of $\{x^{(k)}\}_{k\ge 0}$ is an L-stationary point.*

Proof: Let x^* be an accumulation point of the sequence of $\{x^{(k)}\}_{k\ge 0}$. Since the set of K-sparse vectors is closed, any such x^* is K-sparse. If x^* is an accumulation point, then there exists a subsequence $\{x^{(k_r)}\}_{r\ge 0}$ such that $\lim_{r\to\infty} x^{(k_r)} = x^*$. By Lemma B.4, we also have $\lim_{r\to\infty}\|x^{(k_r)} - x^{(k_r+1)}\|^2 = 0$. Combing these two statements, we can conclude that $\lim_{r\to\infty} x^{(k_r+1)} = x^*$.

Consider the non-zero components of x^*, i.e., components with $x_i^* \ne 0$. Since both $x^{(k_r)}$ and $x^{(k_r+1)}$ converge to x^*, there exists an R such that $x_i^{(k_r)}, x_i^{(k_r+1)} \ne 0$, for all $r \ge R$. Therefore, for $r \ge R$,

$$x_i^{(k_r+1)} = x_i^{(k_r)} - \tfrac{1}{L}\left(\nabla_i f\left(x^{(k_r)}\right) + \epsilon_i^{(k_r)}\right). \quad (25)$$

Since $\sum_{k=0}^{\infty}\|\epsilon^{(k)}\|^2$ is bounded, we have $\lim_{k\to\infty}\epsilon_i^{(k)} = 0$. Thus, taking r to ∞ in (25), we obtain that $\nabla_i f(x^*) = 0$.

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

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

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

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

Taking r to infinity and noting that $\epsilon^{(k_r)} \to 0$, we obtain,

$$\left|x_i^* - \tfrac{1}{L}\nabla_i f(x^*)\right| \le \mathcal{M}_K(x^*),$$

or equivalently, $|\nabla_i f(x^*)| \le L\mathcal{M}_K(x^*)$. Therefore, by Lemma B.2, x^* is an L-stationary point, proving the theorem. ∎

Theorem 4.2 (restated) *Let $f(x) = \|Ax - b\|_2^2$, with spark(A) $> K$. 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 $\sum_{k=0}^{\infty} \|\epsilon^{(k)}\|_2^2 < \infty$. 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 B.3, we have

$$\sum_{s=0}^{k-1} f(x^{(s)}) - f(x^{(s+1)}) \geq \sum_{s=0}^{k-1} \tfrac{L-L_f}{2} \|x^{(s)} - x^{(s+1)}\|^2 - \sum_{s=0}^{k-1} \|x^{(s)} - x^{(s+1)}\| \|\epsilon^{(s)}\| \quad (26)$$

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

By the assumption on the square summability of the error terms, there exists $E < \infty$ such that $E = \sum_{s=0}^{k-1} \|\epsilon^{(s)}\|^2$. We also define $0 \leq F < \infty$ such that $F = \sum_{s=0}^{k-1} \|x^{(s)} - x^{(s+1)}\|^2$. Note that, by Lemma B.4, such an F exists. With these definitions, we arrive at the following inequality for $k \geq 0$,

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

Let T be the level set,

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

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 spark$(A) > K$, then the set T is bounded (see [10], Theorem 3.2). Thus the sequence $\{x^{(k)}\}_{k\geq 0}$ is bounded.

Using an argument identical to one in the proof of Theorem 3.2 in [10], it can be shown that $\{x^{(k)}\}_{k\geq 0}$ converges to an L-stationary point. We repeat this argument 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 [10], 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 there exists an $\epsilon_1 > 0$ such that for all $J \geq 0$, there exists a $j > J$ 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,

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

Since, by assumption, $\{x^{(k)}\}_{k\geq 0}$ does not converge to x^*, each t_j is well-defined. Given the definition of $x^{(t_j)}$ and the fact that $\|x^{(k)} - x^{(k+1)}\| \to 0$ (by Lemma B.4), there exists a subsequence of $x^{(t_j)}$ that converges to a point z^* with $\|x^* - z^*\| \leq \epsilon$. 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 C
Pseudocode for Diffusive Distributed Consensus

Algorithm 3: Diffusive Distributed Consensus algorithm for agent p.

```
initialize
    if $p = 1$ then
        initiated ← TRUE
    else
        initiated ← FALSE
    activeNeighbors ← $\emptyset$
    $v_p^{(0)}$ ← initial value
for $t = 0 \ldots \infty$ do
    if initiated = TRUE then
        $\mathcal{N}_p(t)$ ← 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 ← activeNeighbors ∪ $\{q\}$
    if receive INITIATE from some agent $q$ then
        if initiated = FALSE then
            initiated ← TRUE
        activeNeighbors ← activeNeighbors ∪ $\{q\}$
```

Pseudocode for the diffusive distributed consensus algorithm is given Algorithm 3.

Appendix D
Pseudocode for Other Recovery Algorithms

The pseudocode for D-ADMM is given in Algorithm 4. The pseudocode for the distributed subgradient algorithm is given in Algorithm 5. In the distributed subgradient algorithm, $[\ \cdot\ ]^{+}_{A_p x_p - b_p}$ denotes the projection onto the constraint set $A_p x_p = b_p$.

Appendix E
Simulation Results for Broadcast Communication

We present evaluation results for the convergence of the different algorithms using a broadcast model of communication in a static network. We use the same evaluation setting as in Section V-C. We note that the time evaluation results in Section V-C also apply to a broadcast setting since they account for messages being sent on an agent's links in parallel.

When an agent broadcasts a message, it is sent to all of its neighbors in the original graph. In the broadcast phase of a DIHT iteration, when an agent broadcasts a message, all of its neighbors receive the message, but only the agent's children process this message; the others discard it. Similarly, in the convergecast phase, while all neighbors of an agent receive a

TABLE VI: Total number of broadcasts needed for convergence to an accuracy of 10^{-2} in a static network.

(a) Sparco problem 902.

Graph	DIHT	D-ADMM	CB-DIHT	Subgrad.
BA	2.31×10^6	3.97×10^6	5.10×10^7	$>1.00 \times 10^{10}$
ER (*pr*=0.25)	2.31×10^6	5.24×10^6	7.89×10^7	1.99×10^9
ER (*pr*=0.75)	2.30×10^6	8.70×10^6	4.18×10^7	9.54×10^8
Geo (*d*=0.5)	2.31×10^6	4.43×10^6	2.48×10^8	1.91×10^9
Geo (*d*=0.75)	2.30×10^6	6.85×10^6	6.05×10^7	8.50×10^8

(b) Sparco problem 7.

Graph	DIHT	D-ADMM	CB-DIHT	Subgrad.
BA	6.03×10^6	1.71×10^7	3.31×10^8	$>2.56 \times 10^{10}$
ER (*pr*=0.25)	6.02×10^6	2.07×10^7	3.36×10^8	$>2.56 \times 10^{10}$
ER (*pr*=0.75)	6.02×10^6	2.52×10^7	3.37×10^8	9.83×10^8
Geo (*d*=0.5)	6.03×10^6	1.65×10^7	1.08×10^9	$>2.56 \times 10^{10}$
Geo (*d*=0.75)	6.02×10^6	3.40×10^7	3.31×10^8	2.45×10^9

(c) Sparco problem 11. For DIHT with $L = 4750$ and CB-DIHT with $L_{TV} = 4570/P$, in the vast majority of experiments, the algorithms converge to an L-stationary point that is not the original signal. The values shown for DIHT and CB-DIHT are for convergence to the L-stationary point; these values are preceded by a †. For convergence to the original signal, the values in these columns would all be infinite. For all other columns, the values shown are for convergence to the original signal.

Graph	DIHT	DIHT	D-ADMM	CB-DIHT	CB-DIHT	Subgradient
	$L = 4570$	$L = 500$		$L_{TV} = 4570/P$	$L_{TV} = 500/P$	
BA	†3.18×10^7	1.49×10^6	1.52×10^7	†4.33×10^9	8.14×10^7	$>1.31 \times 10^{10}$
ER (*pr*=0.25)	†3.18×10^7	1.49×10^6	3.45×10^7	†4.32×10^9	7.84×10^7	$>1.31 \times 10^{10}$
ER (*pr*=0.75)	†3.18×10^7	1.48×10^6	1.04×10^8	†1.08×10^9	7.41×10^7	3.34×10^9
Geo (*d*=0.5)	†3.18×10^7	1.49×10^6	2.28×10^7	†6.31×10^8	4.46×10^9	$>1.31 \times 10^{10}$
Geo (*d*=0.75)	†3.18×10^7	1.48×10^6	8.04×10^7	†2.34×10^9	7.85×10^7	1.11×10^9

Algorithm 4: 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
        $u_p^{(k)} \leftarrow \gamma_p^{(k)} - \rho \sum_{\substack{q \in \mathcal{N}p \\ col(q) < c}} x_q^{(k+1)} - \rho \sum_{\substack{q \in \mathcal{N}p \\ col(q) > c}} x_q^{(k)}$
        $x_p^{(k+1)} \leftarrow \arg\min_{x_p} \frac{1}{P}\|x_p\|_1 + u_p^{(k)\mathsf{T}} x_p + \frac{D_p}{2}\|x_p\|_2^2$
                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 5: 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 $x_p^{(k)}$
    $x_p^{(k+1)} \leftarrow \left[ u_p^{(k)} - \alpha^{(k)} g_p^{(k)} \right]^{+}_{A_p x_p = b_p}$
```

broadcast message, only the parent of that agent processes it. In CB-DIHT, D-ADMM, and the subgradient method, each message is broadcast to all of the agents' neighbors in the original graph, and they all process that message. In all algorithms, each broadcast contains a single value. So, to send an N-vector, N broadcasts are needed, and to send a K-vector, $2K$ broadcasts are needed.

For each algorithm, we measure the number of broadcasts needed for $\|x_p^{(t)} - x^*\|/\|x^*\|$ to be less than 10^{-2} at all agents. As in Section V, for D-ADMM and the subgradient algorithm, x^* is the original sparse signal from the Sparco toolbox. For DIHT and CB-DIHT, x^* is the relevant L-stationary point. We clearly indicate when x^* is not the original signal in the evaluation results. For each experiment, we ran the simulation until convergence within the desired accuracy or for 2×10^5 iterations, whichever occurred first.

The simulation results are given in Table VI. In DIHT the total number of broadcasts per iteration depends on the topology of the spanning tree, since leaf nodes do not have any children to which to broadcast the iterate. The topology of the tree, in turn, depends on the topology of the original graph. Therefore, while the total number of broadcasts is similar for each graph for a given problem, it is not identical. For Sparco problems 902 and 7, DIHT requires significantly fewer broadcasts than D-ADMM, and this difference is more pronounced in problem 7. CB-DIHT requires an order of magnitude more broadcasts than DIHT and D-ADMM in most cases. and the subgradient algorithm requires at least two orders of magnitude more broadcasts than DIHT and D-ADMM in all cases. For Sparco problem 11, when $L = 4570$, DIHT requires more broadcasts than D-ADMM to converge, and it does not converge to the original signal. When $L = 500$, DIHT recovers the original signal, requiring an order of magnitude fewer broadcasts than D-ADMM to do so. Both CB-DIHT and the subgradient algorithm require significantly more broadcasts to achieve convergence.

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*, Mar 2009, pp. 181–185.

[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*, Jun 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. Communications*, May 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, Mar 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.*, May 2011, pp. 2916–2919.

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

[8] ——, “D-ADMM: A communication-efficient distributed algorithm for separable optimization,” *IEEE Trans. Sig. Proc.*, vol. 61, no. 10, pp. 2718–2723, May 2013.

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

[10] A. Beck and Y. C. Eldar, “Sparsity constrained nonlinear optimization: Optimality conditions and algorithms,” *SIAM J. Optimiz.*, vol. 23, no. 3, pp. 1480–1509, Oct 2013.

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

[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.*, Jun 2009.

[13] D. Jakovetic, J. Xavier, and J. Moura, “Cooperative convex optimization in networked systems: Augmented lagrangian algorithms with directed gossip communication,” *IEEE Trans. Sig. Proc.*, vol. 59, no. 8, pp. 3889–3902, Aug 2011.

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

[15] 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, Oct 2011.

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

[17] A. I. Chen and A. E. Ozdaglar, “A fast distributed proximal-gradient method,” in *Proc. of Allerton Conf. on Communication, Control, and Computing*, Oct 2012, pp. 601–608.

[18] 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.*, Dec. 2011, pp. 1458–1467.

[19] 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.

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

[21] 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.*, May 2013, pp. 4494–4498.

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

[23] V. D. Blondel, J. M. Hendrickx, A. Olshevsky, and J. N. Tsitsiklis, “Convergence in multiagent coordination, consensus, and flocking,” in *Proc. Joint 44th IEEE Conf. Decision and Control and European Control Conf.*, Dec 2005, pp. 2996–3000.

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

[25] E. J. Candès, “Compressive sampling,” in *Proc. Int. Congr. Math.*, Aug 2006, pp. 1433–1452.

[26] A. Segall, “Distributed network protocols,” *IEEE Trans. Inf. Theory*, vol. 29, no. 1, pp. 23–35, Jan 1983.

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

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

[29] 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, Oct 2007.

[30] A. Barabási and R. Albert, “Emergence of scaling in random networks,” *Science*, vol. 286, no. 5439, pp. 409–410, Oct 1999.

[31] P. Erdös and A. Rényi, “On random graphs I.” *Publ. Math. Debrecen*, vol. 6, pp. 290–297, 1959.

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

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

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

[35] H. Baala, O. Flauzac, J. Gaber, M. Bui, and T. El-Ghazawi, “A self-stabilizing distributed algorithm for spanning tree construction in wireless ad hoc networks,” *J. Parallel Distrib. Comput.*, vol. 63, no. 1, pp. 97–104, Jan 2003.

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

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