MODIFIED DISTRIBUTED ITERATIVE HARD THRESHOLDING

Puxiao Han, Ruixin Niu

Virginia Commonwealth University
Dept. of Electrical and Computer Engineering
Richmond, VA, 23284, U.S.A.
Email: {hanp, rniu}@vcu.edu

Yonina C. Eldar

Technion-Israel Institute of Technology
Dept. of Electrical Engineering
Haifa, 32000, Israel
Email: yonina@ee.technion.ac.il

ABSTRACT

In this paper, we suggest a modified distributed compressed sensing (CS) approach based on the iterative hard thresholding (IHT) algorithm, namely, distributed IHT (DIHT). Our technique improves upon a recently proposed DIHT algorithm in two ways. First, for sensing matrices with i.i.d. Gaussian entries, we suggest an efficient and tight method for computing the step size μ in IHT based on random matrix theory. Second, we improve upon the global computation (GC) step of DIHT by adapting this step to allow for complex data, and reducing the communication cost. The new GC operation involves solving a Top-K problem and is therefore referred to as GC.K. The GC.K-based DIHT has exactly the same recovery results as the centralized IHT given the same step size μ. Numerical results show that our approach significantly outperforms the modified thresholding algorithm (MTA), another GC algorithm for DIHT proposed in previous work. Our simulations also verify that the proposed method of computing μ renders the performance of DIHT close to the oracle-aided approach with a given "optimal" μ.

Index Terms— Distributed Compressed Sensing, Iterative Hard Thresholding, Communication Cost

1. INTRODUCTION

With the exponential growth of sensor data, it becomes challenging for compressed sensing (CS) [1,2] on a single processor. Hence, distributed CS (DCS) has become an interesting topic. It generally contains two parts: (1) the local computation (LC) performed at each sensor, and (2) the global computation (GC) which gathers data from all the sensors.

Several DCS algorithms have been recently proposed [3–11]. In [3], a distributed subspace pursuit (DiSP) algorithm was developed to recover joint sparse signals. In DiSP, each sensor stores the global sensing matrix, and the LC step involves optimization and matrix inversion. The computation and memory burden may become very challenging for each sensor in large-scale problems. In [4], an algorithm named distributed alternating direction method of multipliers (D-ADMM) based on basis pursuit (BP) was proposed, in which sensors do not have to store the entire global sensing matrix. However, each sensor still needs to solve an optimization problem per iteration, and to broadcast its solution to its neighbors. This typically results in high communication cost since the recovery in the first few iterations is not sparse.

To address these problems, a DCS algorithm based on the iterative hard thresholding (IHT) [12, 13] algorithm, named D-IHT was proposed in [5] and [6]. In the LC, each sensor performs very simple operations such as matrix transpose, addition and multiplication. The GC step uses a modified thresholding algorithm (TA) [14], which is a popular method to solve the distributed Top-K problem in the field of database querying. The modification reduces the amount of messages sent between sensors. D-IHT requires computing a step size as part of the IHT algorithm, which ideally should be chosen as $\alpha/||A||_2$. Here A is the CS sensing matrix, $||A||_2$ denotes its largest singular value, and $\alpha \in (0, 1)$ is a scaling parameter close to 1. Exact computation of $\|A\|_2$ requires at least one sensor to have access to the global sensing matrix. To relax this assumption, an upper bound on the norm is developed in [6] which depends on the norms of the local sensing matrices. However, this approximation leads to a much more conservative step size and induces a low convergence rate. Furthermore, the modified TA (MTA) proposed in [5] can only be applied to real-valued CS recovery.

In this paper, we develop a new version of Distributed IHT (DIHT), in which these two issues are addressed. First, we propose a statistical approach to obtain a tight upper bound on $\|A\|_2$, which only depends on the number of rows and columns of A; second, we propose a new Top-K algorithm, which is named GC.K, to accomplish the GC in DIHT in both real-valued and complex-valued cases. As demonstrated later by numerical results, the proposed modified DIHT significantly outperforms the MTA-based DIHT.

We use the following definitions and notations in this paper: $A\backslash B$ denotes the set difference between A and B; the cardinality of a set S is denoted by $|S|$. $v(k)$ denotes the k-th component of the vector v. $[\cdot]^T$ and $[\cdot]^H$ denote the transpose and conjugate transpose of a matrix or vector respectively. $\|\cdot\|_0$ denotes the number of non-zero components of a vector.

978-1-4673-6997-8/15/$31.00 ©2015 IEEE

2. MODIFIED DIHT ALGORITHM

2.1. The Centralized IHT Algorithm

The goal of IHT is to recover an unknown K-sparse vector $s_0 \in \mathbb{C}^N$ given its measurement $y = As_0 + e$, where e is noise, and the sensing matrix $A \in \mathbb{C}^{M \times N}$:

$$x_{t+1} = \eta(x_t + \mu A^H(y - Ax_t); K) \quad (1)$$

where μ is a step size within $(0, 1/\|A\|_2)$, and $\eta(v; K)$ for $v \in \mathbb{C}^n$ is a hard thresholding function, which returns a K-sparse vector $u \in \mathbb{C}^n$ computed by

$$u(k) = \begin{cases} v(k) & \text{if } |v(k)| \geq \mathcal{T}_K(v) \\ 0 & \text{otherwise} \end{cases}, \ \forall k = 1, \cdots, n, \quad (2)$$

where $\mathcal{T}_K(\cdot)$ is the K-th largest absolute component of a vector. In this paper, we draw all the entries of A from independent and identically distributed (i.i.d.) $\mathcal{N}(0, 1/M)$ so that x_t in (1) can converge to a value x^* close to s_0 with a linear convergence rate [12, 13, 15], with high probability.

2.2. The GC.K Algorithm

In a distributed sensor network with P distributed sensors, each sensor p has M/P rows of A, denoted as A^p, takes a measurement $y^p = A^p s_0 + e^p$, and computes

$$w_t^p = \begin{cases} x_t + \mu(A^p)^H(y^p - A^p x_t), & p = 1 \\ \mu(A^p)^H(y^p - A^p x_t), & \text{otherwise} \end{cases} \quad (3)$$

Then the original IHT algorithm can be rewritten as [5, 6]

$$x_{t+1} = \eta\left(\sum_{p=1}^{P} w_t^p; K\right) \quad (4)$$

It can be shown that the communication happens at GC of x_{t+1}. Let $f_t = \sum_{p=1}^{P} w_t^p$, according to (2), $x_{t+1}(n) = 0$ if $|f_t(n)| < \mathcal{T}_K(f_t)$. Therefore, we only need to know all $(n, f_t(n))$ such that $|f_t(n)| = |\sum_{p=1}^{P} w_t^p(n)| \geq \mathcal{T}_K(f_t)$ in the GC. This is a Top-K problem, in which the n-th row of $W_t := \left[w_t^1, \cdots, w_t^P\right]$ can be viewed as an object with index n and partial scores $w_t^1(n), \cdots, w_t^P(n)$ stored on agents (sensors) $1, \cdots, P$, respectively, and the total score of object n is $f_t(n) = \sum_{p=1}^{P} w_t^p(n)$. The objective is to find the K largest-in-magnitude total scores $f_t(n) = \sum_{p=1}^{P} w_t^p(n)$, as well as the indices n of objects they correspond to at a minimal communication cost.

In previous work [5], the MTA algorithm was proposed to solve this problem. However, as will be shown later, MTA becomes inefficient when K and the number of sensors P are large, or when the signal to noise ratio is low; furthermore, it cannot be applied to the complex-valued CS. Another popular Top-K algorithm is the three-phase uniform threshold (TPUT) approach [16]. Despite the fact that it is not directly applicable in our case since it requires all the entries in W_t to be non-negative, a basic idea of TPUT, namely upper bounding the total scores, is the basis for our proposed Top-K algorithm, referred to as GC.K, which is shown in Table 1, where the parameter θ is to trade off the communication cost in Step II and Step III. We use the symbol '$\star$' indicating that communication occurs thereafter in the paper. It is easy to show that the number of messages in GC.K is $\sum_{p=2}^{P} |\Omega_p \bigcup F| + |F| + 1$.

By applying the triangular inequality, it can be shown that in each iteration t, $U(n)$ and $L(n)$ in the GC.K are upper and lower bounds on $|f_t(n)|$ respectively. Furthermore, ν_3 in step III is equal to $\mathcal{T}_K(f_t)$ and GC.K gives exactly the same x_{t+1} as that computed by (1). Fig. 1 gives an example of GC.K with $K = 2$ and $\theta = 0.8$, which consumes 14 messages.

Table 1. GC.K algorithm

Input $w_t^1, \cdots, w_t^P, K, \theta$

Step I Define $\Omega_p^1 := \{n : |w_t^p(n)| \geq T_K(w_t^p)\}$ for each p;
for sensor p = 2:P
$\star$ send all $(n, w_t^p(n))$ pairs for $n \in \Omega_p^1$ to Sensor 1.
endfor
Sensor 1 defines R_n and $P(n)$, $\forall n \in \bigcup_{p=1}^{P} \Omega_p^1$ as follows:
$R_n = \{p : n \in \Omega_p^1\}$ and $P(n) = \sum_{p \in R_n} w_t^p(n)$;
Define F^1 as the set of indices of the K largest $|P(n)|$'s;
$\star$ Sensor 1 broadcasts F^1 to other sensors;
for sensor p = 2:P
$\star$ send all $(n, w_t^p(n))$ pairs for $n \in F^1 \backslash \Omega_p^1$ to Sensor 1;
endfor
Sensor 1 computes $f_t(n)$ for each $n \in F^1$;
Let ν_1 be the K-th largest element in $\{|f_t(n)| : n \in F^1\}$;
Step II $\star$ Sensor 1 broadcasts ν_1 to other sensors;
for sensor p = 2:P
Set $T = \nu_1 \theta/(P-1)$;
define $\Omega_p^2 := \{n : |w_t^p(n)| > T\} \backslash (\Omega_p^1 \bigcup F_1)$;
$\star$ send all $(n, w_t^p(n))$ pairs for $n \in \Omega_p^2$ to Sensor 1.
define $\Omega_p := \Omega_p^1 \bigcup \Omega_p^2 \bigcup F_1$;
endfor
Sensor 1 defines S_n, $L(n)$ and $U(n)$, $\forall n \notin F^1$ as follows:
$S_n := \{p \geq 2 : n \in \Omega_p\}$;
$L(n) = \min\{0, |w_t^1(n) + \sum\limits_{p \in S_n} w_t^p(n)| - (P - 1 - |S_n|)T\}$;
$U(n) = |w_t^1(n) + \sum_{p \in S_n} w_t^p(n)| + (P - 1 - |S_n|)T$;
Let ν_2 be the K-th largest $L(n)$, and $\nu = \max\{\nu_1, \nu_2\}$;
Define $F^2 := \{n \notin F^1 : U(n) \geq \nu\}$;
Step III $\star$ Sensor 1 broadcasts F^2 to other sensors;
for sensor p = 2:P
$\star$ send all $(n, w_t^p(n))$ pairs for $n \in F^2 \backslash \Omega_p$ to sensor 1.
endfor
Sensor 1 computes $f_t(n)$ for all $n \in F^2$;
Define $F := F^1 \bigcup F^2$;
Let ν_3 be the K-th largest element in $\{|f_t(n)| : n \in F\}$;
Define $\Gamma = \{n \in F : |f_t(n)| \geq \nu_3\}$;
Assign $x_{t+1}(n) = f_t(n)$,$\forall n \in \Gamma$ and $x_{t+1}(n) = 0$,$\forall n \notin \Gamma$;

Output x_{t+1}

From the mechanism of GC.K, it is clear that GC.K is applicable to both real-valued and complex-valued cases. In contrast, the MTA proposed in [5], which is shown in Table 2 and also returns exactly the same x_{t+1} as in (1), requires each sensor to sort the partial scores (not by magnitudes). This only works if all the data are real valued.

For evaluating the communication cost, considering the approach sending all the data to Sensor 1, which has a total number of messages $N(P-1)$, we use the ratio between the number of messages of GC.K and $N(P-1)$, denoted as μ_M, to measure the efficiency of GC.K. After Sensor 1 obtains x_{t+1}, it needs K messages to broadcast the non-zero components in x_{t+1} to other sensors. So we also define $T_M = \mu_M + K/[N(P-1)]$ to evaluate the performance of GC.K-based DIHT.

For MTA, as shown in Table 2, in each for-loop iteration inside the while-loop, the algorithm consumes $P+1$ messages, and there are totally N_s such iterations. So the number of messages in MTA is $N_s(P+1)$. It can be shown that if we run MTA on the data in Fig. 1, then we will get $N_s = 9$, which corresponds to $9 \times (3+1) = 36$ messages. After MTA terminates, each sensor has obtained the same x_{t+1}, hence there is no additional broadcasts for the non-zero components of x_{t+1}. Since the communication cost is proportional to $N_s \leq N$, we define μ_M for the MTA as $\mu_M = N_s/N$, and $T_M = N_s(P+1)/[N(P-1)]$. Note that the definitions of μ_M in GC.K and MTA are slightly different.

2.3. The step size μ in DIHT

In centralized IHT, we set μ close to $1/\|A\|_2$ in pursuit of a considerable convergence rate. However, the exact computation of $\|A\|_2$ needs access to the global sensing matrix, which contradicts the basic assumption of the DCS framework.

An alternative proposed in [6] is to obtain an upper bound on $\|A\|_2$. Each sensor $p \geq 2$ computes and sends $\|A^p\|_2$ to Sensor 1. Sensor 1 then computes $L_U = \sum_{p=1}^{P} \|A^p\|_2^2$, which is an upper bound on $\|A\|_2^2$, sets $\mu = 1/\sqrt{L_U}$, and broadcasts μ to the other sensors. However, L_U is generally a loose upper bound on $\|A\|_2^2$, leading to a much smaller μ than the centralized IHT.

Here, we propose a new approach DIHT.S, which provides a better approximation of μ, by applying random matrix theory (RMT). Let $G = AA^T$ and $L_1 = \|G\|_2$. Then $\|A\|_2 = \sqrt{L_1}$. By [17], if $A := [a_{ij}]_{M\times N}$ with $a_{ij} \sim$ i.i.d. $\mathcal{N}(0, 1/M)$, then in the large system limit ($N \to \infty$ and $M/N \to \kappa > 0$),

$$L_1 \xrightarrow{\mathcal{D}} \mu_{MN} + \sigma_{MN} T_1 \text{ with } T_1 \sim F_1 \tag{5}$$

where

$$\mu_{MN} = (1 + \sqrt{(N-1)/M})^2\,, \tag{6}$$

$$\sigma_{MN} = \frac{\sqrt{M} + \sqrt{N-1}}{M} \left(\frac{1}{\sqrt{M}} + \frac{1}{\sqrt{N-1}} \right)^{1/3}, \tag{7}$$

and F_1 in (5) is the cumulative distribution function of the Tracy-Widom law of order 1 [18], with standard deviation 1.27. By (7), in the large system limit, the standard deviation of L_1 approaches $1.27\sigma_{MN} \to 0$, implying that L_1 will become more and more "deterministic". Hence we can obtain a statistical upper bound $L(\alpha) = \mu_{MN} + \sigma_{MN} F_1^{-1}(1-\alpha)$ (α is a smaller number, and in the simulations we set $\alpha = 0.01$), which is the approximate $(1-\alpha)$-th quantile for L_1. Due to the fact that $\sigma_{MN} \to 0$, this bound will be very tight. We then set $\mu = 1/\sqrt{L(\alpha)}$. Note that each sensor can calculate μ which only depends on M and N, without data transmission.

Table 2. MTA Algorithm

Input $w_t^1, \cdots, w_t^P$, K

Initialize $x_{t+1} = 0 \in \mathbb{R}^N$, $\texttt{count}= 0$, $\tau_T = +\infty$, $\tau_B = +\infty$,
  $u_p = +\infty$, $\ell_p = -\infty$, $\forall p = 1, \cdots, P$;
Mark all the pairs $(n, w_t^p(n))$ as "unsent", $\forall n, p$;
while $\texttt{TRUE}$
  for sensor p = 1:P
    obtain $R = \{n : (n, w_t^p(n))$ is marked as "unsent"$\}$;
    if $\tau_T \geq \tau_B$
      set $n_s = \arg\max_{n\in R} w_t^p(n)$;
      update $u_p = w_t^p(n_s)$ and $\tau_T = \max\{0, \sum_{q=1}^{P} u_q\}$;
    else
      set $n_s = \arg\min_{n\in R} w_t^p(n)$;
      update $\ell_p = w_t^p(n_s)$ and $\tau_B = -\min\{0, \sum_{q=1}^{P} \ell_q\}$;
    endif
    $\star$ broadcast $(n_s, w_t^p(n_s))$ and mark it as "sent";
    for sensor $q \neq p$
    $\star$ send $(n_s, w_t^q(n_s))$ to sensor p and mark it as "sent";
      store $w_t^p(n_s)$ as the new u_p or ℓ_p;
    endfor
    $\star$ compute $f_t(n_s)$ and broadcast it to other sensors;
    $\texttt{count=count+1}$;
    let β be K-th largest element in $\{|f_t(n)| : n \notin R\backslash\{n_s\}\}$;
    if $\max\{\tau_T, \tau_B\} < \beta$ or $\texttt{count} \geq N$
      update $x_{t+1}(n) = f_t(n)$ if $|f_t(n)| > \beta, \forall n \notin R\backslash\{n_s\}$;
      set $N_s = \texttt{count}$, the algorithm terminates;
    endif
  endfor
endwhile

Output x_{t+1}

---

3. NUMERICAL RESULTS

We fix $N = 5000$, set $M = N\kappa$ and $K = M\rho$, where $\kappa \in \{0.2, 0.3, 0.4, 0.5\}$ and $\rho \in \{0.1, 0.15, 0.2, 0.25\}$, and choose $P \in \{10, 15, \cdots, 50\}$. s_0 is generated with random support and non-zero components drawn from $\mathcal{N}(0, 1)$. The noise $e \sim \mathcal{N}(0, \sigma^2 I_M)$ with $\sigma \in \{0.01, 0.02, \cdots, 0.09\}$. IHT terminates if $\|x_{t+1} - x_t\|_2 \leq 0.001\|x_t\|_2$ or if it runs up to 100 iterations. θ in GC.K is set to 0.8. We have the following setup: i) fix $(P, \sigma) = (10, 0.02)$, and change (κ, ρ); ii) fix $(\kappa, \rho, P) = (0.2, 0.1, 10)$, and change σ; iii) fix $(\kappa, \rho, \sigma) = (0.2, 0.1, 0.02)$, and change P. Under each parameter setting,

Sensor 1 $(n, w_t^1(n))$	Sensor 2 $(n, w_t^2(n))$	Sensor 3 $(n, w_t^3(n))$	Step I $(n, P(n))$	Step I $(n, f_t(n))$	Step II Ω_p^2, Ω_p $p \geq 2$	Step II $(n, L(n))$	Step II $(n, U(n))$	Step III $(n, f_t(n))$	Step III $(n, x_{t+1}(n))$
(6, 9) (4, -8) (7, -8) (5, 6) (2, 3) (9, -3) (3, 2) (1, -1) (8, 0) (10, 0) $\Omega_1^1 = \{6,4\}$	(6, 10) (2, -7) (4, 7) (8, -5) (9, -5) (1, 4) (5, 4) (7, -3) (10, -3) (3, 1) $\Omega_2^1 = \{6,2\}$	(1, 10) (7, -10) (3, -9) (5, -9) (4, 8) (8, 7) (10, -5) (6, 4) (2, -2) (9, 0) $\Omega_3^1 = \{1,7\}$	(6, 19) (7, -18) (1, 9) (4, -8) (2, 4) $\bigcup_{p=1}^{3} \Omega_p^1 = \{6,$ 7,1,4,2} $F^1 = \{6,7\}$	(6, 23) (7, -21) $v_1 = 21,$ $T = \frac{\theta v_1}{P-1}$ $= 8.4$	$\Omega_2^2 = \varnothing$ $\Omega_2 = \{6,2,$ 7} $\Omega_3^2 = \{3,5\}$ $\Omega_3 = \{1,7,$ 3,5,6}	(1, 0) (2, 0) (3, 0) (4, 0) (5, 0) (8, 0) (9, 0) (10, 0) $v_2 = 0$ $v = \max\{v_1, v_2\}$ $= 21$	(4, 24.8) (9, 19.8) (8, 16.8) (10, 16.8) (1, 16.4) (3, 15.4) (5, 11.4) (2, 9.4) $F^2 = \{4\}$	(4, 7) (6, 23) (7, -21) $F = \{4,6,7\}$ $v_3 = 21$	(1, 0) (2, 0) (3, 0) (4, 0) (5, 0) (6, 23) (7, -21) (8, 0) (9, 0) (10, 0)

Fig. 1. An example of GC.K algorithm with $P = 3$, $K = 2$ and $\theta = 0.8$.

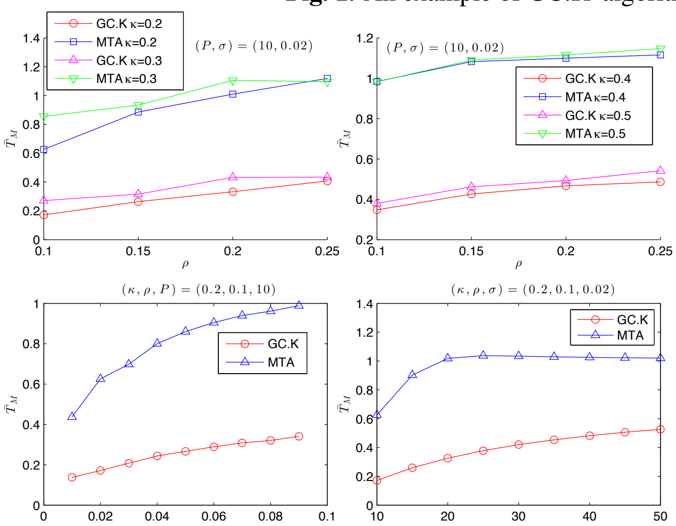

Fig. 2. Communication cost of GC.K and MTA.

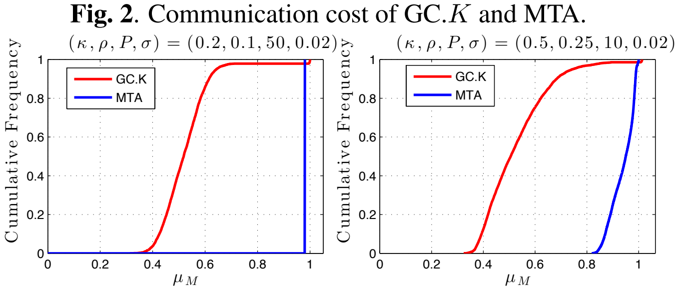

Fig. 3. Cumulative distributions of μ_M for GC.K and MTA.

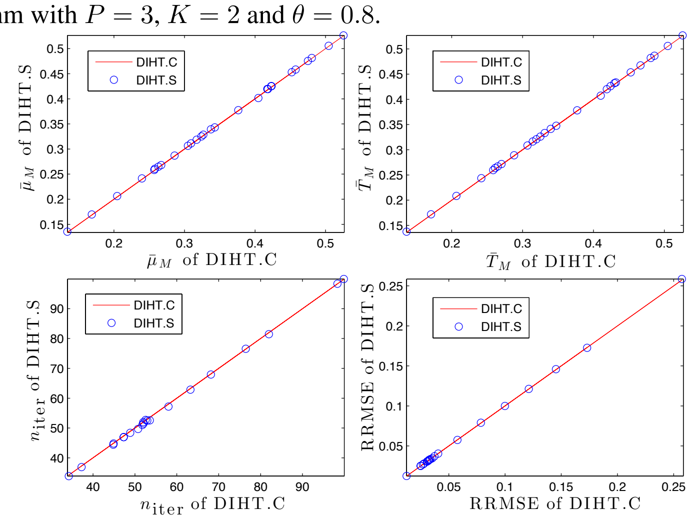

Fig. 4. Comparison of DIHT.S and DIHT.C.

we take $n_{\text{sim}} = 100$ Monte-Carlo runs.

We first compare the GC.K-based DIHT.S and MTA-based DIHT.S. Since they have the same recovery results, we only compare their communication cost, i.e., μ_M and T_M defined at the end of Section 2.2..

Fig. 2 shows $\bar{T}_M$, the sample mean of T_M's, obtained by the two algorithms. As σ, P and K increase, the values of $\bar{T}_M$ in MTA become close to 1, which means that MTA hardly saves any communication cost, while GC.K can still work efficiently. In all the cases, GC.K outperforms MTA. Fig. 3 depicts the cumulative distributions of μ_M for GC.K and MTA under two extreme settings (large P and large K). In all iterations under these two settings, the number of messages in MTA are greater than $0.8N(P-1)$, while GC.K can save at least $0.35N(P-1)$ messages in 80% of the total iterations.

Next, we compare GC.K-based DIHT.S with the oracle-aided approach GC.K-based DIHT.C, where $\|A\|_2$ is known and $\mu = 0.99/\|A\|_2$. The recovery accuracy is measured in terms of relative root mean squared error (RRMSE), which is defined as $\frac{\sqrt{\sum_{i=1}^{n_{\text{sim}}} ||(x_i^* - s_0)||_2^2 / n_{\text{sim}}}}{||s_0||_2}$, where x_i^* is the recovery of the i-th Monte-Carlo run. The convergence rate is evaluated in terms of $\bar{n}_{\text{iter}} := \sum_{i=1}^{n_{\text{sim}}} n_{\text{iter}}^i / n_{\text{sim}}$, where n_{iter}^i is the number of iterations in the i-th Monte-Carlo run. Fig. 4 shows these quantities as well as the communication cost of DIHT.S and DIHT.C respectively, under all parameter settings, where $\bar{\mu}_M$ denotes the sample mean of μ_M's. As we can see, DIHT.S performs similarly to DIHT.C.

We also observe the ratios $\bar{\mu}_M / \bar{T}_M$ for GC.K under all parameter settings, and find that they are within the interval $[0.9771, 0.9989]$, which shows that GC.K takes most of the communication cost in the corresponding DIHT algorithms.

4. CONCLUSION

In this paper, we propose a new distributed IHT approach. For the computation of the step size, we propose a statistical approach DIHT.S which provides a very tight statistical upper bound on $\|A\|_2$ that only depends on the dimensionality of A. In the global computation stage, we propose a new Top-K algorithm GC.K, which outperforms MTA proposed in an earlier work, and renders the corresponding DIHT algorithm applicable to complex-valued compressed sensing.

5. REFERENCES

[1] J. A. Tropp and S. J. Wright, "Computational methods for sparse solution of linear inverse problems," *Proceedings of the IEEE*, vol. 98, no. 6, pp. 948–958, 2010.

[2] M. F. Duarte and Y. C. Eldar, "Structured compressed sensing: From theory to applications," *Signal Processing, IEEE Transactions on*, vol. 59, no. 9, pp. 4053–4085, 2011.

[3] D. Sundman, S. Chatterjee, and M. Skoglund, "A greedy pursuit algorithm for distributed compressed sensing," in *Proc. IEEE Int. Conf. on Acoust., Speech, and Sig. Proc. (ICASSP)*, 2012, pp. 2729–2732.

[4] J. Mota, J. Xavier, P. Aguiar, and M. Puschel, "Distributed basis pursuit," *IEEE Trans. Sig. Proc.*, vol. 60, pp. 1942–1956, April 2012.

[5] S. Patterson, Y. C. Eldar, and I. Keidar, "Distributed sparse signal recovery for sensor networks," in *Proc. IEEE Int. Conf. on Acoust., Speech, and Sig. Proc. (ICASSP)*, 2013, pp. 4494–4498.

[6] S. Patterson, Y. C. Eldar, and I. Keidar, "Distributed compressed sensing for static and time-varying networks," *IEEE Trans. Sig. Proc.*, vol. 62, no. 19, pp. 4931–4946, Oct 2014.

[7] P. Han, R. Niu, M. Ren, and Y. C. Eldar, "Distributed approximate message passing for sparse signal recovery," in *Signal and Information Processing (GlobalSIP), 2014 IEEE Global Conference on*. IEEE, 2014, pp. 497–501.

[8] S. Chouvardas, K. Slavakis, Y. Kopsinis, and S. Theodoridis, "A sparsity promoting adaptive algorithm for distributed learning," *Signal Processing, IEEE Transactions on*, vol. 60, no. 10, pp. 5412–5425, 2012.

[9] S. Chouvardas, G. Mileounis, N. Kalouptsidis, and S. Theodoridis, "A greedy sparsity-promoting lms for distributed adaptive learning in diffusion networks," in *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*. IEEE, 2013, pp. 5415–5419.

[10] P. Di Lorenzo and A. H. Sayed, "Sparse distributed learning based on diffusion adaptation," *Signal Processing, IEEE Transactions on*, vol. 61, no. 6, pp. 1419–1433, 2013.

[11] J. Chen, Z. J. Towfic, and A. H. Sayed, "Online dictionary learning over distributed models," in *Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on*. IEEE, 2014, pp. 3874–3878.

[12] T. Blumensath and M. E. Davies, "Iterative thresholding for sparse approximations," *Journal of Fourier Analysis and Applications*, vol. 14, no. 5-6, pp. 629–654, 2008.

[13] T. Blumensath and M. E. Davies, "Iterative hard thresholding for compressed sensing," *Appl. Comput. Harmon. Anal.*, vol. 27, pp. 265–274, November 2008.

[14] R. Fagin, A. Lotem, and M. Naor, "Optimal aggregation algorithms for middleware," in *Symposium on Principles of Database Systems*, 2001, pp. 614–656.

[15] E. J. Candes, "Compressive sampling," in *Int. Congress of Mathematicians*, Madrid, Spain, 2006, vol. 3, pp. 1433–1452.

[16] P. Cao and Z. Wang, "Efficient top-k query calculation in distributed networks," in *Intl. Symposium on Principles Of Distributed Computing (PODC)*, 2004, pp. 206–215.

[17] I. M. Johnstone, "On the distribution of the largest eigenvalue in principal components analysis," *The Annals of Statistics*, vol. 29, no. 2, pp. 295–327, 04 2001.

[18] C. A Tracy and H. Widom, "Level-spacing distributions and the airy kernel," *Communications in Mathematical Physics*, vol. 159, no. 1, pp. 151–174, 1994.