Distributed Approximate Message Passing for Compressed Sensing

Puxiao Han, Ruixin Niu, and Mengqi Ren
Department of Electrical and Computer Engineering
Virginia Commonwealth University, Richmond, VA, 23284, U.S.A.
Email: \{hanp, niu, renm\}@vcu.edu

Abstract—In this paper, an efficient distributed approach for implementing the approximate message passing (AMP) algorithm, named distributed AMP (DAMP), is developed for compressed sensing (CS) recovery in sensor networks with the sparsity $K$ unknown. In the proposed DAMP, distributed sensors do not have to use or know the entire global sensing matrix, and the burden of computation and storage for each sensor is reduced. To reduce communications among the sensors, a new data query algorithm, called global computation for AMP (GCAMP), is proposed. The proposed GCAMP based DAMP approach has exactly the same recovery solution as the centralized AMP algorithm, which is proved theoretically in the paper. The performance of the DAMP approach is evaluated in terms of the communication cost saved by using GCAMP. For comparison purpose, thresholding algorithm (TA), a well known distributed Top-K algorithm, is modified so that it also leads to the same recovery solution as the centralized AMP. Numerical results demonstrate that the GCAMP based DAMP outperforms the Modified TA based DAMP, and reduces the communication cost significantly.

Index Terms—Compressed Sensing, Distributed AMP.

I. INTRODUCTION

Compressed sensing (CS) has wide applications in various areas of signal processing [1]. Due to the curse of dimensionality, it can be highly demanding to perform CS on a single processor. Further, distributed processing has the potential to reduce communications among distributed sensors. Hence, distributed CS (DCS) in sensor networks has become an interesting topic. A general DCS system contains two parts: (1) the local computation performed at each sensor, and (2) the global computation to obtain the estimate of the original sparse signal after sensors exchange the results of local computation. Several distributed approaches based on various CS recovery algorithms were proposed. In [2], a distributed subspace pursuit (DiSP) algorithm was developed to recover joint sparse signals. In DiSP, each sensor needs to store the global sensing matrix, and local computation at each sensor involves optimization and matrix inversion. The computation and memory burden may become very challenging for each sensor in large-scale problems. Further, in DiSP the sparsity $K$ is assumed to be known, which may not be the case in many applications. In [3], an algorithm named 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 to get an recovery per iteration, and broadcasts it to its neighbors, which may induce high communication cost since the recovery in first few iterations is not sparse. Focusing on these problems, a DCS algorithm based on iterative hard thresholding (IHT) named D-IHT was proposed in [4]. In the local computation, each sensor just performs very simple operations such as matrix transpose, addition and multiplication. In the global computation, thresholding algorithm (TA) [5] has been applied, which is a popular method to solve the distributed Top-K problem in the field of database querying, to reduce the amount of messages sent between sensors. Nevertheless, in the D-IHT, the sparsity $K$ was also assumed to be known. Further, the D-IHT requires each local sensor to know certain prior knowledge about the global sensing matrix, such as its $L_2$ norm. For a certain sensor node (or a fusion center) to know the global sensing matrix to calculate and then broadcast its $L_2$ norm, each of the rest sensor nodes has to either transmit its local sensing matrix or the seed of its local random number generator used to generate the corresponding local sensing matrix.

In this paper, we do not assume the knowledge of sparsity and hence the IHT cannot be directly applied. Instead, we propose a distributed algorithm based on approximate message passing (AMP) [6], which does not require any prior knowledge of the sparse signal, and has a linear convergence rate [6]. [7]. For the proposed distributed AMP (DAMP) approach, we do not assume any prior knowledge of the global sensing matrix. Distributed sensors do not need to store the entire global sensing matrix. In the local computation, each sensor node (or a fusion center) to know the global sensing matrix to calculate and then broadcast its $L_2$ norm, each of the rest sensor nodes has to either transmit its local sensing matrix or the seed of its local random number generator used to generate the corresponding local sensing matrix.

II. DAMP SYSTEM

A. The Original AMP

A task of CS is to recover a sparse signal $s_0 \in \mathbb{R}^N$ from its measurement $y = As_0 + n$, where $A \in \mathbb{R}^{M \times N}$ is the sensing matrix and $n$ is an additive noise, by solving the problem:

$$\min_x \frac{1}{2}||y - Ax||_2^2 + \lambda ||x||_1$$

where $\lambda > 0$ is the regularization parameter. However, $\lambda$ is not given in most practical cases. AMP is a good solution to the
problem [6] without prior knowledge about $K$ and $\lambda$. Starting from $x_0 = 0$ and $z_0 = y$, it recursively gets the new estimate of $s_0$ as follows:

$$x_{t+1} = \eta_t (x_t + A^T z_t; \tau \sigma_t)$$  \hspace{1cm} (2)

$$z_{t+1} = y - Ax_{t+1} + \frac{||x_{t+1}||_0}{M} z_t$$  \hspace{1cm} (3)

where $[\cdot]^T$ denotes the transpose operation, $|| \cdot ||_0$ is the $l_0$ norm of a vector, $\sigma_t^2 = \frac{||z_t||^2}{M}$ [8].

\[ \eta_t(x; \beta) = \begin{cases} (|x| - \beta) \text{sgn}(x), & |x| > \beta \\ 0, & |x| \leq \beta \end{cases} \hspace{1cm} (4) \]

and $\tau$ is a parameter whose optimal value depends on $\kappa = \frac{M}{\rho}$ and $\rho = \frac{1}{N}$ [8]. Since $K$ is unknown, a tuning procedure is needed, which will be presented later in this paper, to find a value for $\tau$ which is very close to the optimum.

B. The Distributed Framework of AMP

Let us consider a sensor network with $P$ distributed sensors. Each sensor $p$ ($p = 1, \cdots, P$) takes a measurement of $s_0$ as

$$\begin{bmatrix} y^1 \\ \vdots \\ y^P \end{bmatrix} = \begin{bmatrix} A^1 \\ \vdots \\ A^P \end{bmatrix} s_0 + \begin{bmatrix} n^1 \\ \vdots \\ n^P \end{bmatrix}$$  \hspace{1cm} (5)

Then, (2) and (3) can be re-written as:

$$x_{t+1} = \eta_t \left( x_t + \sum_{p=1}^P A^p \tau \sigma_t \right)$$  \hspace{1cm} (6)

$$z_{t+1}^p = y^p - A^p x_{t+1} + \frac{||x_{t+1}||_0}{M} z_t, \forall p = 1, \cdots, P$$  \hspace{1cm} (7)

By introducing an intermediate matrix $W_t = [w_1, \cdots, w_P]$ with each column computed by the corresponding sensor as:

$$w_t^p = \begin{cases} x_t + A^p \tau \sigma_t, & p = 1 \\ A^p z_t, & \text{otherwise} \end{cases}$$  \hspace{1cm} (8)

which is similar to that in [4], (5) becomes

$$x_{t+1} = \eta_t \left( \sum_{p=1}^P w_t^p \tau \sigma_t \right)$$  \hspace{1cm} (9)

Therefore, DAMP can be divided into two parts: local computation of $z_t^p$ and $w_t^p$ ($p = 1, \cdots, P$), and global computation of $x_{t+1}$ and $\sigma_t$, in which transmission of data between sensors is needed. For the latter, a natural approach is to send all the data in $w_t^p$ ($p = 2, \cdots, P$) to sensor 1, which induces a high communication cost when $N$ is large. Therefore, how to reduce the communication cost, meanwhile maintaining the same recovery solution as the centralized AMP, is the main focus of this paper.

C. GCAMP Algorithm

Let us denote $v(n)$ as the $n$-th component of a vector $v$. According to (2), $x_{t+1} = 0$ if $|\sum_{p=1}^P w_t^p|^2 \leq \beta = \tau \sigma_t$. Therefore, we only need to know all the $n$s such that $|\sum_{p=1}^P w_t^p| > \beta$ in the global computation. This is similar to Top-K problem in the field of distributed database querying, which is to find the $K$ largest components of $\sum_{p=1}^P w_t^p$. In [9] the three-phase uniform threshold (TPUT) algorithm, an efficient approach to solve the Top-K problem with a known $K$, is proposed. However, our problem is different from the Top-K problem. First, we do not know how many components of $\sum_{p=1}^P w_t^p$ have magnitude larger than $\beta$; second, TPUT requires $w_t^p$’s to be non-negative, while in our problem, they can be any real numbers. Hence, TPUT cannot be applied in our case. Nevertheless, it does provide some insight on how to design the communication algorithm in distributed systems. Here, we propose the GCAMP algorithm which is shown in Table I

| TABLE I |
|-----------------|
| **GCAMP Algorithm** |
|-----------------|
| Input $w_1^1, \cdots, w_t^P, \beta = \tau \sigma_t$ |
| Step I Set $T = \theta \theta/(P - 1)$, where $\theta$ is tuned parameter; for sensor $p = 2: P$ |
| denote $R_p = \{ n : |w_t^p(n)| > \beta \}$; |
| send all $(n, w_t^p(n))$ pairs for $n \in R_p$ to sensor 1; |
| endif |
| Step II for sensor $1$, define $I_S(x) := 1$ if $x \in S$; $0$ o.w.; |
| for $n = 1: N$ |
| get $S_n := \{ p = 2, \cdots, P : I_{R_p}(n) = 1 \}$ with cardinality $m_n$; |
| Compute $U(n) = |w_t^1(n) + \sum_{p \in S_n} w_t^p(n)| + (P - 1 - m_n)T$; |
| if $U(n) > \beta$ and $m_n < P - 1$ |
| broadcast the index $n$ to other sensors; |
| endif |
| Step III denote $F = \{ n : U(n) > \beta, m_n < P - 1 \}$; |
| for sensor $p = 2: P$ |
| send all $(n, w_t^p(n))$ pairs for $n \in F \setminus R_p$ to sensor 1; |
| endif |
| Step IV for sensor $1$, initialize $x_{t+1} = 0$; |
| for $n \in V := \{ n : U(n) \geq \beta \}$ |
| Update $x_{t+1}(n) = \eta_t \left( \sum_{p=1}^P w_t^p(n); \beta \right)$ by (4); |
| endif |
| Output $x_{t+1}$ |

**Theorem 1:** In each iteration, $U(n)$ is an upper bound of $|\sum_{p=1}^P w_t^p(n)|$ for all $n$, and the $x_{t+1}$ which GCAMP algorithm obtains (denoted as $x_{t+1}^{GCA}$) is exactly the same as that obtained by the original centralized AMP algorithm (denoted as $x_{t+1}$).  

**Proof:** For any $n = 1, \cdots, N$, we have

$$\sum_{p=1}^P w_t^p(n) = w_t^1(n) + \sum_{p \in S_n} w_t^p(n) + \sum_{p \geq 2, p \notin S_n} w_t^p(n)$$  \hspace{1cm} (10)

Then, applying the triangle inequality, we have

$$|\sum_{p=1}^P w_t^p(n)| \leq |w_t^1(n) + \sum_{p \in S_n} w_t^p(n)| + |\sum_{p \geq 2, p \notin S_n} w_t^p(n)|$$  \hspace{1cm} (11)

$$\leq |w_t^1(n) + \sum_{p \in S_n} w_t^p(n)| + (P - 1 - m_n)T = U(n)$$
in Table II

How to choose the maximum candidate value, i.e., $\tau_{\text{max}}$, is another interesting problem. In [10], the authors set $\tau_{\text{max}} = \sqrt{||u||_2^2/M}$, which $|| \cdot ||_2$ is the magnitude of the largest-in-magnitude component in a vector. Denote $\tilde{x}_t := x_t + A^Tz_t = \Sigma_{p=1}^{P}u_p^T$, since at the beginning, $x_0 = 0$ and $z_0 = y$, we have $\tilde{x}_0 = A^Ty$ and $s_0 = \sqrt{\frac{||y||_2^2}{M}}$. Therefore, $\forall n = 1, \cdots, N$, we have $|\tilde{x}_n(n)| \leq s_{\text{max}}\sigma_0$. This implies that the optimal value for $\tau$ cannot be greater than $\tau_{\text{max}}$. Here, we propose a different approach. According to [11], as $N \rightarrow \infty$, asymptotically each component of $\tilde{x}_n - s_0$ is independent and identically distributed (i.i.d.) random variable, following a $N(0, \sigma_n^2)$ distribution. Therefore, we can build a $(1 - \alpha)$ confidence interval (CI) $[-z_0\sigma_0, z_0\sigma_0]$, where $z_0$ is defined such that $\frac{1}{\sqrt{2\pi}} \int_{z_0}^{\infty} \exp(-t^2/2) \, dt = \alpha$. Hence, $\forall n = 1, \cdots, N$, if $s_0(n) = 0$, with probability $1 - \alpha$, $\tilde{x}_n(n)$ will be in the CI; on the other hand, if for some $n$, $|\tilde{x}_n(n)| > z_0\sigma_n$, then with probability at least $1 - \alpha$, $s_0(n)$ is a non-zero component. Therefore, we can choose a very small $\alpha$, and let $\tau_{\text{max}} = z_0\sigma_n$. For example, we can let $\alpha = 0.0027$ and $\tau_{\text{max}} = z_0\sigma_n = 3$. Note that in every iteration involving (6) and (7), after GCAMP returns $x_{t+1}$, sensor 1 broadcasts non-zero components of $x_{t+1}$ as well as their indices. In DAMP, we tune the optimal $\tau$ value in a descending order, which implies a larger threshold $\beta = \sigma_1$ in the beginning. Therefore, different from [3], we have a sparse estimate $x_{t+1}$ even at the first few iterations. Hence, the communication cost for broadcasting $x_{t+1}$ is negligible compared with that of GCAMP. Once knowing $x_{t+1}$, each local sensor can obtain $z_{p,t+1}$ using (7) and $\sigma_{p,t+1}^2 = ||u_{p,t+1}||_2^2$. Next, each sensor $p \geq 2$ just sends a scalar $u_{p,t+1}$ to sensor 1, which needs $P - 1$ messages. Then, sensor 1 computes $\sigma_{t+1} = \sqrt{\Sigma_{p=1}^{P}(\sigma_{p,t+1}^2)/M}$, updates $\beta$ and $T$, and broadcasts the scalar $T$ to other sensors. Overall, GCAMP incurs most of the communication cost in DAMP.

E. Comparison of GCAMP and Modified TA

TA [5] is another popular algorithm solving Top-K problems. Similar to TPUT, TA also requires the knowledge of $K$ and all entries in $W_t$ to be non-negative. Therefore, we propose a modified TA algorithm as in Table III and let it be a control algorithm for GCAMP.

Theorem 2: In each iteration, Modified TA algorithm also gives exactly the same $x_{t+1}$ as that of original AMP algorithm.

Proof: Modified TA is composed of a series of global summation, where a global summation means computing $\Sigma_{p=1}^{P}w_p(n)$ for some $n$. $N_s$ is a counter recording the number of global summations. At the very end of one global summation, for each $n$, either the $(n, w_p(n))$ pairs for all $p$ are marked as “sent”; or none of them are marked as “sent”. So we can just say $n$ is marked as “sent” or not. It is easy to show that, $\Sigma_{p=1}^{P}||u_p||_2$ is an upper bound of $\Sigma_{p=1}^{P}w_p(n)$ for all $n$ that have not been marked as “sent”; if $\Sigma_{p=1}^{P}||u_p||_2 \leq \beta$, then we have $\Sigma_{p=1}^{P}w_p(n) \leq \beta$ for these $n$. As the algorithm terminates, we do not lose any non-zero components of $x_{t+1}$.

---

D. Tuning of $\tau$ Values

With the GCAMP algorithm, DAMP can be developed. We adopt the tuning framework in [10] to find the optimal value for $\tau$. First, a descending candidate list of candidate values of $\tau$, $\{\tau(l)\}_{l=1}^{L} := [\tau_{\text{max}}, \cdots, \tau_{\text{max}} - (l - 1)\Delta\tau, \cdots, \tau_{\text{max}} - (L - 1)\Delta\tau]$ is generated. Then, for each candidate $\tau_l$, we run iterations in (6) and (7) until $x_1$ and $s_0$ converge to $x_1^*$ and $s_0^*$, and use them as the initial estimates for the iterations using the next candidate $\tau_{l+1}$. We repeat this process until $\sigma_0^*$ is not decreasing, and get the optimal $\tau$ value as well as the final estimate of $s_0$. The pseudo code of DAMP algorithm is shown in Table II.
we define normalized message number (NMN) as performance measure, which is widely used in literature \cite{5}, for GCAMP. So we use the number of messages transmitted as an important measure to evaluate the communication cost saved by using DAMP. Instead, as DAMP is a distributed algorithm, it is not necessary to evaluate them again in accuracy and convergence of AMP has been well studied in literature\cite{25}\cite{26}. For GCAMP, the total number of messages is $\sum_{p=1}^{P}|R_p|+|F|+\sum_{p=1}^{P}|F\setminus R_p|$; for Modified TA, in each global summation, there are 1 broadcasting message from sensor to others and $P-1$ incoming messages, so the total number of messages is $PN_s$. It is easy to check that, for the data set in Figure\ref{fig:figure1}, Modified TA needs $PN_s = 3 \times 9 = 27$ messages, more than twice of that of GCAMP.

### III. NUMERICAL RESULTS

#### A. Performance Measures

Since we have proved that the DAMP algorithm has exactly the same solution as the original AMP, and the recovery accuracy and convergence of AMP has been well studied in literature, it is not necessary to evaluate them again in this paper. Instead, as DAMP is a distributed algorithm, it is important to evaluate the communication cost saved by using GCAMP. So we use the number of messages transmitted as the performance measure, which is widely used in literature \cite{25}, \cite{26}. We compare the number of messages used in GCAMP to that in Modified TA. Considering the approach sending all data to sensor 1, which has a total number of messages $N(P-1)$, we define normalized message number (NMN) as

$$\mu_M = \frac{\text{number of messages in computing } x_{t+1}}{N(P-1)}$$

(12)

which is $\mu_M = \frac{\sum_{i=1}^{P}|R_p|+|F|+\sum_{p=1}^{P}|F\setminus R_p|}{N(P-1)}$ for GCAMP and $\mu_M = \frac{N_p \times P}{N(P-1)}$ for Modified TA.

#### B. Simulation Setup

Our focus is not to investigate large-scale problems, but to develop distributed algorithms and evaluate their efficiency in reducing communication costs. Nevertheless, we still use a considerably large $N = 5000$, and choose $\kappa$ from $[0.1, 0.5]$, $\rho$ from $[0.1, 0.3]$, which leads to $M = N \times K$ in $[500, 2500]$ and $K = M \rho$ in $[50, 750]$. The problem scales used in our paper is larger than those used in other DCS publications \cite{25}. The number of sensors $P$ is within $[5, 50]$. The sensing matrix $A$ with i.i.d. entries $\sim \mathcal{N}(0, \sigma^2 I_M)$ is partitioned into $P$ parts with each sensor having a $(M/P) \times N$ submatrix. Each component of $s_0$ is i.i.d. drawn from

$$f_X(x) = \kappa \rho G(x) + (1 - \kappa \rho) \delta(x)$$

(13)

where $G(x)$ is the probability density function (pdf) of the standard Gaussian distribution and $\delta(x)$ is the Dirac Delta function. The measurements of $s_0$ are corrupted by an additive noise $n \sim \mathcal{N}(0, \sigma^2 I_M)$ and $\sigma$ is the standard deviation with a value in $[0.01, 0.1]$. The parameter $\theta$ in GCAMP is set to 0.8. Regarding the tuning procedure for optimal $\tau$ values, we make a candidate list for $\tau$ of length 11, starting from 3 with a step -0.2; for each candidate, the convergence criteria is $|\sigma_\tau - \sigma_{t-1}| < 0.01 |\sigma_{t-1}|$. We compare $\mu_M$ defined as $\mu_M$ averaged over iterations based on 100 Monte-Carlo runs.

#### C. Performance Evaluation

We evaluate $\mu_M$ in three settings: I) fix $\sigma = 0.02$ and $P = 10$, and change the values of $\kappa$ and $\rho$; II) fix $\kappa = 0.2$, $\rho = 0.1$ and $P = 10$, and change the values of $\sigma$; III) fix $\kappa = 0.2$,

| TABLE II
| DAMP ALGORITHM |
|---|
| **Input** $(y)_p^P, (A)_p^P, \{\tau\}_{i=1}^L$, maximer, $\epsilon$; |
| **Initialization** $x_0 = 0$, $z_0^p = y^p$ for $p = 1 \ldots P, \sigma_0 = \sqrt{\sum_{p=1}^{P}||z_0^p||_2^2/M}$; for $i = 1:L$
| for $t = 1$ maximer
| for $p = 1:P$
| Compute $w_1^p = y^p$ by (9); endfor |
| for $p = 1:P$
| Compute $z_0^p$ by (10); endfor |
| $\sigma_1 = \sqrt{\sum_{p=1}^{P}||z_0^p||_2^2/M}$ if $|\sigma_1 - \sigma_{t-1}| < c_\sigma |\sigma_{t-1}|$
| $\sigma(t_i) = \sigma(t_i) = x(t_i), z^p(t_i) = z_0^p$ for $p = 1 \ldots P$; break; endif 
| $\sigma_0 = \sigma(t_i), x_0 = x(t_i), z_0^p = z^p(t_i)$ for $p = 1 \ldots P$; endfor |
| **Output** $x_{t+1}$ |

| TABLE III
| MODIFIED TA ALGORITHM |
|---|
| **Input** $w_1^1, \ldots, w_1^p, \beta = \tau \sigma_M$; |
| **Initialization** $x_{t+1} = 0, N_s = 0; for sensor p = 1:P$
| sort components of $w_1^p$ in descending order of magnitudes; define the sorted vector as $s_1^p$ and $I_1^p(n) := \{i \text{ s.t. } w_1^p(i) = s_1^p(n)\}; mark all $(I_1^p(n), s_1^p(n))$ pairs as "unsent"; endfor while 1
| for $p = 1:P$, do the following process named global summation find the first $(I_1^p(n), s_1^p(n))$ pair marked "unsent" from top; set $u_p = s_1^p(n)$, broadcast $(I_1^p(n), u_p)$ to other sensors; mark $(I_1^p(n), u_p)$ as "sent"; for sensor $q \neq p$
| store $u_p$ and send $(I_1^p(n), u_p'(I_1^p(n)))$ to sensor $p$; mark $(I_1^p(n), u_p'(I_1^p(n)))$ as "sent"; endfor update $x_{t+1}(I_1^p(n)) = \eta_t (\sum_{p=1}^{P} w_1^p(I_1^p(n)))$; 
| number of global summations $N_s = N_s + 1$; if $N_s \geq P$ and $\sum_{p=1}^{P}|u_p| \leq \beta$, or if $N_s \geq N$
| the algorithm terminates; endif 
| endfor 
| endwhile |
| **Output** $x_{t+1}$ |
\[ \rho = 0.1 \text{ and } \sigma = 0.02, \text{ and change the values of } P. \text{ Tables } \text{IV and V} \text{ show the corresponding numerical results for I), II) and III) respectively. In the tables, the former entry in each pair inside the parentheses denotes } \mu_M \text{ for GCAMP, and the latter denotes that for Modified TA. It is clear that in each case, GCAMP outperforms Modified TA significantly. Modified TA always uses more messages than } N(P−1) \text{ except for the case } P = 5, \text{ while GCAMP can save the number of messages from } 22.7\% \text{ to } 48.2\%. \text{ Fig. 2 gives the cumulative distributions of } \mu_M \text{ in each iteration for GCAMP and Modified TA under 4 different scenarios: 1) } \kappa = 0.2, \rho = 0.1, \sigma = 0.02, P = 5; \text{ 2) } \kappa = 0.2, \rho = 0.1, \sigma = 0.02, P = 10; \text{ 3) } \kappa = 0.2, \rho = 0.1, \sigma = 0.01, P = 10; \text{ 4) } \kappa = 0.3, \rho = 0.1, \sigma = 0.02, P = 10. \text{ It provides us much more detailed information on the distribution of } \mu_M \text{ for each algorithm. It is clear that under each scenario, Modified TA uses more than } N(P−1) \text{ messages in at least } 33.4\% \text{ of the total iterations; while GCAMP never uses more than } 0.91N(P−1) \text{ messages in any iteration, and among more than } 95\% \text{ of the total iterations, it just uses } \lceil 40\%, 80\% \rceil \times N(P−1) \text{ messages, that is, it can save } 20\% \sim 60\% \text{ of the messages with probability at least } 95\%. \text{ TABLE IV}

| \kappa = 0.1 | 0.2 | 0.3 | 0.4 | 0.5 |
|-------------|-----|-----|-----|-----|
| \rho=0.10   | (0.547, 1.101) | (0.567, 1.103) | (0.573, 1.103) | (0.587, 1.103) | (0.589, 1.103) |
| 0.15        | (0.621, 1.108) | (0.661, 1.106) | (0.632, 1.107) | (0.635, 1.107) | (0.639, 1.106) |
| 0.20        | (0.659, 1.108) | (0.667, 1.106) | (0.672, 1.107) | (0.691, 1.109) | (0.684, 1.108) |
| 0.25        | (0.651, 1.107) | (0.689, 1.109) | (0.704, 1.109) | (0.725, 1.109) | (0.731, 1.109) |
| 0.30        | (0.632, 1.108) | (0.690, 1.109) | (0.737, 1.109) | (0.751, 1.110) | (0.755, 1.110) |

| \sigma = 0.01 | 0.02 | 0.03 | 0.04 | 0.05 |
|---------------|-----|-----|-----|-----|
| (0.564, 1.103) | (0.567, 1.103) | (0.574, 1.104) | (0.576, 1.104) | (0.582, 1.104) |
| (0.583, 1.104) | (0.589, 1.104) | (0.590, 1.104) | (0.592, 1.105) | (0.590, 1.105) |
| \sigma = 0.06 | 0.07 | 0.08 | 0.09 | 0.1 |
| (0.583, 1.104) | (0.589, 1.104) | (0.590, 1.104) | (0.592, 1.105) | (0.590, 1.105) |

| \mu_M \text{ for GCAMP and Modified TA with different } \sigma |
|-----|-----|-----|-----|-----|

| \mu_M \text{ for GCAMP and Modified TA with different } P |
|-----|-----|-----|-----|

IV. Conclusion

Assuming the sparsity of the original signal to be unknown, the DAMP approach has been developed for performing compressed sensing in distributed sensor networks, consisting a series of local and global computations. We proposed the

REFERENCES

[1] M. Duarte and Y. Eldar, “Structured compressive sensing: From theory to applications,” IEEE Trans. Sig. Proc., vol. 59, pp. 4053–4085, September 2011.
[2] 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.
[3] J. Mota, J. Xavier, P. Aguiar, and M. Puschel, “Distributed basis pursuit,” IEEE Trans. Sig. Proc., vol. 60, pp. 1942–1956, April 2012.
[4] S. Patterson, Y. 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.
[5] R. Foglin, A. Lotem, and M. Naor, “Optimal aggregation algorithms for middlewire,” in Symposium on Principles of Database Systems, 2001, pp. 614–656.
[6] D. L. Donoho, A. Maleki, and A. Montanari, “Message passing algorithms for compressed sensing,” in Proc. Natl. Acad. Sci., vol. 106, Madrid, Spain, September 2009, pp. 18914–18919.
[7] A. Maleki and R. G. Baraniuk, “Least favorable compressed sensing problems for the first order methods,” in Proc. IEEE Int. Symp. Inf. Theory, 2011, pp. 134–138.
[8] D. L. Donoho, A. Maleki, and A. Montanari, “The Noise-Sensitivity Phase Transition in Compressed Sensing,” IEEE Trans. Info. Theory, vol. 57, pp. 6920–6941, October 2011.
[9] 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.
[10] L. Anitori, A. Maleki, M. Otten, R. G. Baraniuk, and P. Hoogeboom, “Design and Analysis of Compressed Sensing Radar Detectors,” IEEE Trans. Signal Proc., vol. 61, pp. 813–827, February 2013.
[11] M. Bayati and A. Montanari, “The Dynamics of Message Passing on Dense Graphs, with Applications to Compressed Sensing,” IEEE Trans. Info. Theory, vol. 57, pp. 764–785, February 2011.