A Bayesian Approach for Asynchronous Parallel Sparse Recovery

Alireza Zaeemzadeh[1], Jamie Haddock[2], Nazanin Rahnavard[1], Deanna Needell[2]
[1]School of Electrical Engineering and Computer Science, University of Central Florida
Emails: {zaeemzadeh and nazanin}@eecs.ucf.edu
[2]Department of Mathematics, University of California, Los Angeles
Emails: {jhaddock and deanna}@math.ucla.edu

Asynchronous parallel algorithms are often studied for separable optimization problems where the component objective functions are *sparse*, or act on only a few components of the variable $x \in \mathbb{R}^N$. One challenge to developing asynchronous approaches for sparse recovery is that the optimization formulation of this problem has dense component objective functions. However, the assumed sparsity of the signal may be exploited in an asynchronous parallel approach. Here we propose such an approach where multiple processors asynchronously infer hidden variables that estimate the support of x in a Bayesian manner. We include numerical simulations that demonstrate the potential benefits of this method.

I. Introduction

Sparse recovery problems have received significant attention in the past decade, particularly in the compressed sensing (CS) literature [1, 2]. CS techniques have revolutionized sensing and sampling, with applications in image reconstruction [3, 4], hyper spectral imaging [5], wireless communications [6, 7], and analog to digital conversion [8]. Meanwhile, complex data-gathering devices have been developed, leading to the rapid growth of *big data*. For instance, the size of problems in hyperspectral imaging [5] are so large that they cannot be stored or solved in conventional computers. This, as well as the proliferation of inexpensive multi-processor computing systems, has motivated the study of parallel sparse recovery.

In parallel sparse recovery, the goal is to solve a large-scale sparse recovery problem by partitioning it among multiple processing nodes, thus reducing both the storage and computation requirements [9]. However, many recent studies [9–14] focus on *synchronous* parallel recovery of the sparse signal, meaning that some subset of the processing nodes need to wait for another subset of the nodes to complete their tasks. Of course, this approach is sensitive to slow or nonfunctional nodes.

Thus, it is natural to look for algorithms that divide the large-scale sparse recovery problems among several computing nodes and solve it *asynchronously*. Recently, in [15], the authors proposed a strategy to utilize the stochastic hard thresholding (StoIHT) [16, 17] in an asynchronous manner. Instead of sharing the current solution among the processors, which is the conventional approach [9–11], an estimate of the support of the signal is shared. Then, the iteration number of each processor is used to assign weight to faster cores.

This material is based upon work supported by the National Science Foundation under Grant No. ECCS-1810256, CCF-1718195, Grant No. DMS-1522158, Award No. CAREER-1348721, and Grant No. BIGDATA-1740325, and the University of California, Davis Dissertation Fellowship.

In this paper, inspired by [15], we propose an asynchronous StoIHT [16] which incorporates a probabilistic framework that assigns reliability scores to each processor. This score is calculated by considering both the processor's estimate of the support *and* its iteration number. Therefore, not only do we ignore the information from unreliable slow cores, but we are also able to utilize the reliable information from slower cores and disregard unreliable information from faster cores. The update rules for reliability scores and the support estimation is derived in a mathematical, less heuristic, manner, using variational inference [18]. This leads to simple closed form update rules for the parameters of the posterior distributions of the hidden variables with very low computational overhead.

II. System Model

We consider the *sparse recovery* problem of reconstructing $\boldsymbol{x} \in \mathbb{R}^N$ from few nonadaptive, linear, and noisy measurements, $\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{z}$, where $\boldsymbol{A} \in \mathbb{R}^{m\times N}$ is the measurement matrix and $\boldsymbol{z} \in \mathbb{R}^m$ is noise. One challenge to developing an asynchronous parallel approach to recovering the s-sparse signal $\boldsymbol{x}$ via the optimization problem

$$\min_{\hat{\boldsymbol{x}}\in\mathbb{R}^N} \frac{1}{2m}\|\boldsymbol{y} - \boldsymbol{A}\hat{\boldsymbol{x}}\|_2^2 \quad \text{subject to} \quad \|\hat{\boldsymbol{x}}\|_0 \leq s$$

is that the cost function $\frac{1}{2m}\|\boldsymbol{y} - \boldsymbol{A}\hat{\boldsymbol{x}}\|_2^2$ is defined by the matrix $\boldsymbol{A}$, which is not generally sparse (e.g., standard i.i.d. Gaussian $\boldsymbol{A}$ is common). A naive asynchronous approach would frequently overwrite the s non-zero entries learned by faster and more reliable processors. Our goal is to solve this problem in an asynchronous manner, while reducing the effects of *slow* processors on the estimated signal. Note that the problem can be rewritten as

$$\min_{\hat{\boldsymbol{x}}\in\mathbb{R}^N} \frac{1}{M}\sum_{B=1}^{M}\frac{1}{2b}\|\boldsymbol{y}_B - \boldsymbol{A}_B\hat{\boldsymbol{x}}\|_2^2 \quad \text{subject to} \quad \|\hat{\boldsymbol{x}}\|_0 \leq s,$$

where $\boldsymbol{y}$ and $\boldsymbol{A}$ are partitioned into M non-overlapping sub-vectors $\boldsymbol{y}_B$ and sub-matrices $\boldsymbol{A}_B$. At each iteration, each processor solves a subproblem by using the $b = m/M$ equations defined by $\boldsymbol{A}_B$ and $\boldsymbol{y}_B$. We do not assume that the number of subproblems M and processors P is the same.

III. Probability Model

Our Bayesian algorithmic framework makes use of a *tally vector* $\boldsymbol{\phi} \in \mathbb{R}^N$ which records information on the current

estimated support of $\boldsymbol{x}$. This vector and several reliability estimates are the *hidden variables* in our model:

1) *Tally score*, $\phi_n \in [0,1]$, describing the probability that coefficient n is in support.
2) *Reliability score* for each processor, $r_i \in [0,1]$, denoting the trustworthiness of the measurements of processor i.
3) *Observation reliability*, $u_{ni} \in \{0,1\}$, which indicates if coefficient n reported by processor i is reliable.

Our estimates of these hidden variables are updated according to the following *observed variables*:

1) The *support observations*, o_{ni} which indicates if processor i detects coefficient n in the support.
2) The maximum *number of iterations* completed by any processor since the last reporting of processor i, k_i.

The posterior probability distribution of these hidden variables (referred to as $\mathcal{H}$) are inferred from the observed variables reported by the processors, o_{ni} and k_i (referred to as $\mathcal{D}$), according to the following generative model where variables are indexed for $i = 1, \ldots, P$ and $n = 1, \ldots, N$.

$$\begin{aligned} r_i &\sim \text{Beta}(\beta_i^1, \beta_i^0) \\ u_{ni} &\sim \text{Bernoulli}(r_i) \\ \phi_n &\sim \text{Beta}(a_n^1, a_n^0) \\ o_{ni} &\sim u_{ni}\,\text{Bernoulli}(\phi_n) \\ &\qquad + (1-u_{ni})\,\text{Bernoulli}(1-\phi_n) \\ k_i &\sim \text{Binomial}(K_i, r_i) \end{aligned} \tag{1}$$

The variable for the reliability score, r_i, is modeled with a Beta distribution with parameters β_i^1 and β_i^0. This is the natural choice since r_i is used as the parameter of the Bernoulli distribution of the observation reliability and the conjugate prior for a Bernoulli distribution is Beta distribution.

The observation reliability is modeled as a Bernoulli distribution with parameter r_i. If a processor is generally reliable (r_i close to 1) it is more likely to be reliable on other coefficients.

The tally score is also defined as a random variable sampled from a Beta distribution with parameters a_n^1 and a_n^0. This is also because ϕ_n is later used as the parameter of the Bernoulli distribution that describes the observations.

The observed variable, o_{ni}, is defined as the summation of two Bernoulli distributions. If an observation of the processor is reliable, $u_{ni} = 1$, the distribution would be $\text{Bernoulli}(\phi_n)$. This means that o_{ni} will be sampled from a Bernoulli distribution with true parameter, i.e., ϕ_n. By definition, ϕ_n is defined as the probability that coefficient n belongs to the support of the signal. Otherwise, for $u_{ni} = 0$, it will be sampled from $\text{Bernoulli}(1-\phi_n)$, which means it reports faulty data.

Finally, k_i, the number of iterations completed by processor i is modeled with $\text{Binomial}(K_i, r_i)$ where K_i is the maximum number of iterations completed by any processor since the last reporting of processor i. Thus, we have $k_i \leq K_i$. Reliable processors (r_i close to 1) are likely to report k_i close to K_i.

The goal of our sequential Bayesian updating inference algorithm is to infer the distribution of $\mathcal{H}$ given $\mathcal{D}$.

IV. Inference via Sequential Bayesian Updating

Using Bayes' rule, the posterior distribution is

$$\mathbb{P}\{\mathcal{H}|\mathcal{D}\} \propto \mathbb{P}\{\mathcal{D}|\mathcal{H}\}\mathbb{P}\{\mathcal{H}\} = \mathbb{P}\{\mathcal{D}, \mathcal{H}\}.$$

where $\mathbb{P}\{\mathcal{D}, \mathcal{H}\}$ is calculated using the model described in (1). Specifically, the last two expressions in (1), are used to build $\mathbb{P}\{\mathcal{D}|\mathcal{H}\}$ and the other terms represent our prior belief, $\mathbb{P}\{\mathcal{H}\}$. The posterior distribution is the updated distribution of the hidden variables after receiving the observations.

In sequential Bayesian updating, the prior knowledge of the model is represented as the prior distribution, which is the distribution of the hidden variables before collecting data. After observing the first set of measurements, the posterior distribution is determined using Bayes' rule. Then, the posterior distribution can be used as the prior when the next set of observations becomes available. Thus, we must update the distribution of the hidden variables using the observations. In this approach, all the information is stored in the current distribution of the hidden variables.

To handle the intractable integrals arising in the inference procedure, *variational inference* is employed [18, 19]. In variational inference, the posterior distribution is approximated by a family of distributions for which the calculations are tractable. The approximate distribution is assumed to be fully factorized over all the hidden variables [20, Chapter 10]. Specifically, the fully factorized variational distribution $\mathbb{Q}\{\mathcal{H}\}$ is defined as

$$\mathbb{Q}\{\mathcal{H}\} = \prod_i \mathbb{Q}\{r_i|\hat{\beta}_i^1, \hat{\beta}_i^0\} \prod_{n,i} \mathbb{Q}\{u_{ni}|\tau_{ni}\} \prod_n \mathbb{Q}\{\phi_n|\hat{a}_n^1, \hat{a}_n^0\}, \tag{2}$$

where $\hat{\beta}_n^1$, $\hat{\beta}_n^0$, $\hat{a}_n^1$, $\hat{a}_n^0$, and τ_{ni} are the parameters of the factorized distributions. Our goal is to obtain $\mathbb{Q}\{\mathcal{H}\}$ such that it approximates the posterior distribution $\mathbb{P}\{\mathcal{H}|\mathcal{D}\}$.

Thus, at each step, the optimization problem

$$\max \quad \mathbb{E}\{\ln(\mathbb{P}\{\mathcal{D}, \mathcal{H}\})\} - \mathbb{E}\{\ln(\mathbb{Q}\{\mathcal{H}\})\}$$

(where the expected value is with respect to variational distribution) is solved with respect to one of the factorized distributions, keeping all other distributions fixed. The procedure is repeated until convergence. Each step results in a *closed form* update rule for one of the variables. Since the objective function is convex with respect to each of the factorized distributions, convergence is guaranteed [20, Chapter 10]. The derivations of the updating rules are presented in the Appendix. After receiving each set of new measurements, the probability distributions of the unknown variables are updated using the closed form update rules. In this framework, the tally score for each coefficient, ϕ_n, is a random variable. Thus, the expected value of the random variable is used as a point estimate of the tally score and is denoted by

$$\bar{\phi}_n = \mathbb{E}_{\mathbb{Q}\{\phi_n\}}\{\phi_n\} = \frac{\hat{a}_n^1}{\hat{a}_n^1 + \hat{a}_n^0}. \tag{3}$$

Furthermore, we indicate the tally vector by $\boldsymbol{\phi} = [\bar{\phi}_1, \bar{\phi}_2, \ldots, \bar{\phi}_N]$. Details of the proposed framework can be

found in Algorithm 1 and the performance of the algorithm is evaluated in Section V.

Algorithm 1 Bayesian Asynchronous StoIHT Iteration

Require: Number of subproblems, M, and probability of selection $p(B)$. The parameters of distribution of the reliability score, $\hat{\beta}_i^1$ and $\hat{\beta}_i^0$, and the parameters of tally scores, $\hat{a}_n^1$ and $\hat{a}_n^0$, are available to each processor.

Each processor performs the following at each iteration:

randomize: select $B_t \in [M]$ with probability $p(B_t)$

proxy: $\boldsymbol{b}^{(t)} = \boldsymbol{x}^{(t)} + \frac{\gamma}{Mp(B_t)} \boldsymbol{A}_{B_t}^* (\boldsymbol{y}_{B_t} - \boldsymbol{A}_{B_t} \boldsymbol{x}^{(t)})$

identify: $\hat{\mathcal{S}}^{(t)} = supp_s(\boldsymbol{b}^{(t)})$ and $\tilde{T}^{(t)} = supp_s(\boldsymbol{\phi})$

estimate: $\boldsymbol{x}^{(t+1)} = \boldsymbol{b}^{(t)}_{\hat{\mathcal{S}}^{(t)} \cup \tilde{T}^{(t)}}$

repeat

  update $\mathbb{E}_{\mathbb{Q}\{u_{ni}\}}\{u_{ni}\} = \mathbb{Q}\{u_{ni} = 1\}$ as described in Appendix C

  update $\hat{\beta}_i^1$ and $\hat{\beta}_i^0$ using (9), $\hat{a}_n^1$ and $\hat{a}_n^0$ using (8)

until convergence

update $\boldsymbol{\phi}$ using (3)

$t = t + 1$

It is clear that the inference algorithm is an iterative method. However, we will show in Section V that the framework performs well even if we run the update rules only once.

V. Numerical Experiments

A. Experiments in MATLAB

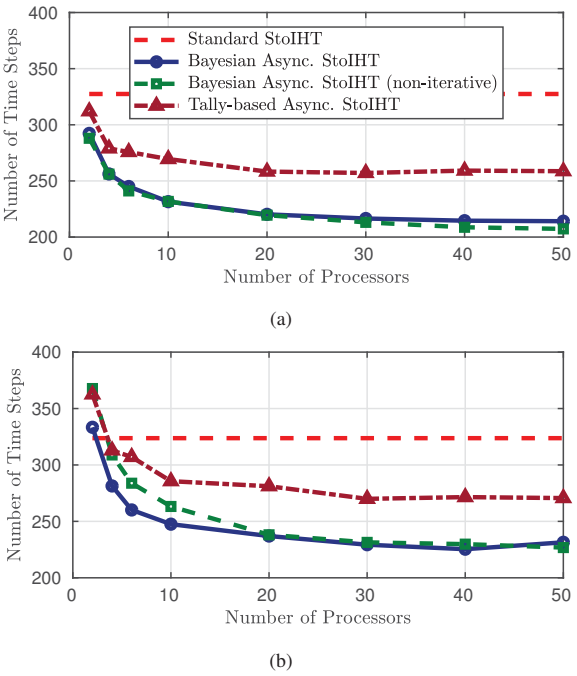
In these experiments, we take the signal dimension $N = 1000$, the sparsity level $s = 20$, and the number of measurements $m = 300$. Also, initial values for β_i^1, β_i^0, a_n^1, and a_n^1 are set to 1, which results in uniform distribution for all r_i and ϕ_n and indicates unbiased estimate of processor reliability and tally score in absence of further information. For Stochastic IHT, the block size b is set to be same as the sparsity level s and $\gamma = 1$. The convergence criteria is $\|\boldsymbol{y} - \boldsymbol{A}\boldsymbol{x}^t\| \leq 10^{-7}$ and the maximum number of iterations is 1500.

Figure 1 shows the mean number of time steps over 50 trials when (a) all processors take the same amount of time to complete an iteration, and (b) half of the processors are *slow*, meaning that they complete an iteration every four time steps. It is evident that time steps required using the Bayesian update rules have decreased compared to standard non-parallel Stochastic IHT and Tally-based asynchronous IHT [15].

As mentioned in Section IV, the proposed inference algorithm is an iterative algorithm and needs to run the update rules alternatively to reach convergence. However, we also evaluate the performance of the non-iterative inference algorithm in which, at each StoIHT iteration, the inference runs the update rule for each variable only once. Figure 1 shows that the non-iterative and iterative algorithms performs similarly.

B. Experiments in C++

In this set of experiments, to have a better understanding of the behavior of the algorithms in a real parallel environment, different sparse recovery methods are implemented and tested



(a)

(b)

Figure 1: Comparison of the number of time steps until convergence versus the number of processors used in different methods, when (a) all processors complete an iteration in a single time step and (a) half of the processors complete an iteration every four time steps.

using the C++ programming language and OpenMP [21], a multiprocessor shared memory programming platform. Here, we take $N = 10000$, $m = 3000$, and $s = 200$. All other simulation parameters are same as Section V-A. Running time reflects the time required to execute all the steps of the algorithms, including initialization, preprocessing, convergence, and post-processing. All simulations have been performed in the Ubuntu 16.04 environment on a PC equipped with an Intel Xeon E5-1650 processor (3.20 GHz) with 12 processors and 8 GB of RAM. Parallel AMP, a synchronous sparse recovery algorithm, is a row-wise multi-processor approximate message passing algorithm, as described in [9]. Here, we use the non-iterative version of the Bayesian asynchronous Sto-IHT algorithm. All the results are based on 50 Monte-Carlo trials.

Figure 2(a) and Figure 2(b) show the execution time per iteration and total convergence time, respectively, for different numbers of processors. In this experiment, slow processors sleep for 100 ms at each StoIHT iteration and make up 20% of the processors; i.e., no slow processors for $P < 5$, one for $5 \leq P < 10$, and two for $P = 10$. After adding the first slow processor, the execution time for parallel AMP increases significantly, illustrating the fact that synchronous parallel algorithms suffer from the presence of slow processors. On the other hand, the execution time of the asynchronous algorithms does not change significantly. It is worthwhile to mention that at $P = 10$, when the second slow processor is added to the system, there is no increase in the execution time of any of

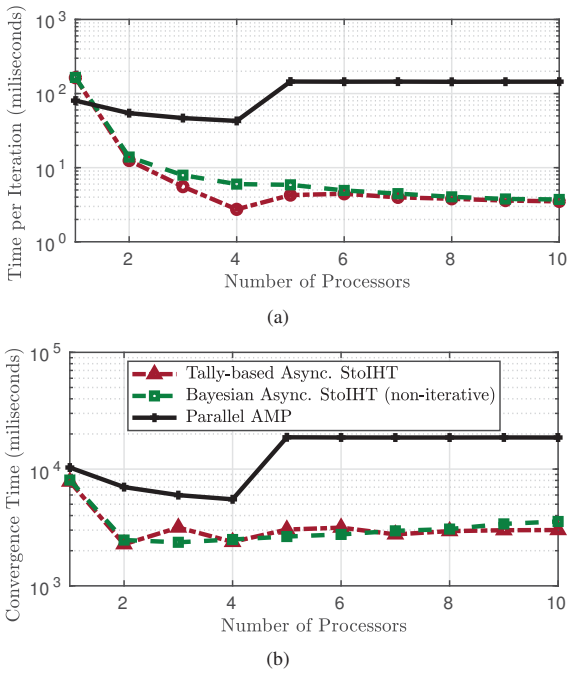


Figure 2: Performance of different multi-processor sparse recovery algorithm implemented using C++ programming language and OpenMP platform. Twenty percent of the processors are slow.

the algorithms, since more slow cores with the same sleep time does not increase the wait time of the system. Figure 2(b) shows that although the execution time per iteration of the asynchronous algorithms is decreasing after adding more processors, the total convergence time increases slightly, since more cores increases the processor/thread scheduling overhead and takes the inference algorithm longer to converge.

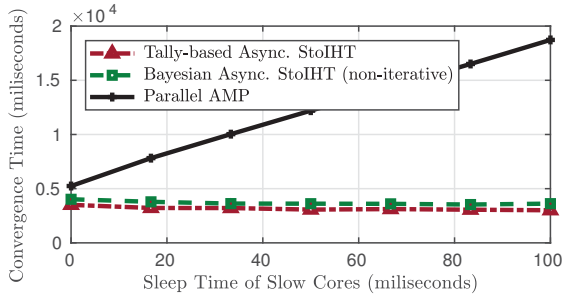


Figure 3: Mean convergence time of different multi-processor sparse recovery algorithms over 50 trials. Ten processors are solving the sparse recovery problem and two cores are slow.

Figure 3 demonstrates the effect of the sleep time of the two slow processors (of ten) on the execution time. Note that the convergence times of synchronous multi-processor algorithms increase linearly as the processors become slower, since all processors need to wait for slower processors at each iteration. The convergence time of the asynchronous algorithms depend less on the sleep time of the slow cores.

VI. Conclusions

In this work, we modified the stochastic iterative hard thresholding algorithm to solve the sparse recovery problem in an asynchronous parallel manner. We proposed a Bayesian framework to assign reliability scores to the processing nodes, using both their current estimate of the support and their iteration number. The update rules for the reliability score and the support estimate are derived in closed form using variational inference. This computationally inexpensive inference makes the algorithm more robust to slow, unreliable processing nodes. An interesting future direction is to utilize this framework for other sparse recovery algorithms such as AMP [22], which is known to have a better phase-transition threshold.

Appendix

In this section, the derivations of the update rules for the inference algorithm are presented. As discussed in Section IV, the posterior distribution is approximated by a family of distributions for which the calculations are tractable, employing the *naive mean field* approach [19].

In (2), $\mathcal{H}$ is divided into disjoint groups $\mathcal{H}_k, k = 1, \ldots,$ where each $\mathcal{H}_k$ is representing one of the hidden variables in $\mathcal{H}$. The variational distribution of each partition $\mathbb{Q}\{\mathcal{H}_k\}$ is given by [20, Chapter 10]

$$\ln(\mathbb{Q}\{\mathcal{H}_k\}) = \mathbb{E}_{j\neq k}\{\ln(\mathbb{P}\{\mathcal{D}, \mathcal{H}\})\} + const, \tag{4}$$

where $\mathbb{E}_{j\neq k}\{.\}$ is the expectation with respect to distributions $\mathbb{Q}\{\mathcal{H}_j\}$. Plugging in $\mathbb{P}\{\mathcal{D}, \mathcal{H}\}$ and using the exponential form of the distributions, we obtain the variational distributions. The constant is determined by normalizing the distribution.

It is worthwhile to state that if $x \sim \text{Bernoulli}(p)$, then

$$\ln(\mathbb{P}\{x\}) = \ln(\frac{p}{1-p})x + \ln(1-p) \tag{5}$$

and if $x \sim \text{Binomial}(n, p)$, we have

$$\ln(\mathbb{P}\{x\}) = \ln(\frac{p}{1-p})x + n\ln(1-p) + \ln(\binom{n}{x}). \tag{6}$$

Also, if $x \sim \text{Beta}(b^1, b^0)$, we have

$$\begin{aligned} \ln(\mathbb{P}\{x\}) &= (b^1-1)\ln(x) + (b^0-1)\ln(1-x) + const \\ \mathbb{E}\{\ln(x)\} &= \psi(b^1) - \psi(b^1+b^0), \\ \mathbb{E}\{\ln(1-x)\} &= \psi(b^0) - \psi(b^1+b^0), \end{aligned} \tag{7}$$

where $\psi(\cdot)$ is the digamma function. We now present the update rules to obtain the approximate posterior distributions.

A. Tally Score

Using (4), (5), (7) and integrating out all variables but ϕ_n, we have

$$\begin{aligned} \ln(\mathbb{Q}\{\phi_n\}) = &\quad \mathbb{E}\{\ln(\mathbb{P}\{\mathcal{D}, \mathcal{H}\})\} + const. \\ = &\quad const + \ln(\mathbb{P}\{\phi_n | a_n^1, a_n^0\}) \\ &\quad + \mathbb{E}_{\mathbb{Q}\{u_{ni}\}}\{\ln(\mathbb{P}\{o_{ni}|u_{ni}, \phi_n\})\} \\ = &\quad const + (a_n^1 - 1)\ln(\phi_n) + (a_n^0 - 1)\ln(1-\phi_n) \\ &\quad + \mathbb{E}_{\mathbb{Q}\{u_{ni}\}}\{u_{ni}\}(\ln(\frac{\phi_n}{1-\phi_n})o_{ni} + \ln(1-\phi_n)) \end{aligned}$$

where i is the updating processor index. The prior knowledge on the tally and (7) provide the first two terms; the last combines processor information, given the observation reliability.

This expression can be further written in the form of $(\hat{a}_n^1 - 1)\ln(\phi_n) + (\hat{a}_n^0 - 1)\ln(1-\phi_n) + const$, which is a Beta distribution with parameters

$$\begin{aligned} \hat{a}_n^1 &= a_n^1 + \mathbb{E}_{\mathbb{Q}\{u_{ni}\}}\{u_{ni}\}o_{ni}, \\ \hat{a}_n^0 &= a_n^0 + \mathbb{E}_{\mathbb{Q}\{u_{ni}\}}\{u_{ni}\}(1-o_{ni}). \end{aligned} \tag{8}$$

Here, $\mathbb{E}_{\mathbb{Q}\{u_{ni}\}}\{.\}$ is expectation with respect to $\mathbb{Q}\{u_{ni}\}$ and $\mathbb{E}_{\mathbb{Q}\{u_{ni}\}}\{u_{ni}\}$ can be calculated using $\mathbb{Q}\{u_{ni}\}$, which will be discussed shortly. This update rule simply means that if $o_{ni} = 1$, we will increase the positive count by $\mathbb{E}_{\mathbb{Q}\{u_{ni}\}}\{u_{ni}\}$; if $o_{ni} = 0$, we will increase the negative count by $\mathbb{E}_{\mathbb{Q}\{u_{ni}\}}\{u_{ni}\}$.

B. Processor Reliability Score

Similarly, to update reliability of each processor i, we have

$$\begin{aligned} \ln(\mathbb{Q}\{r_i\}) &= const + (\beta_i^1 - 1)\ln(r_i) + (\beta_i^0 - 1)\ln(1-r_i) \\ &\quad + \ln(\frac{r_i}{1-r_i})k_i + \ln(1-r_i)K_i \\ &\quad + \sum_n \mathbb{E}_{\mathbb{Q}\{u_{ni}\}}\{\ln(\frac{r_i}{1-r_i})u_{ni} + \ln(1-r_i)\} \\ &= const + \ln(r_i)(\beta_i^1 + \sum_n \mathbb{E}_{\mathbb{Q}\{u_{ni}\}}\{u_{ni}\} + k_i - 1) \\ &\quad + \ln(1-r_i)(\beta_i^0 + \sum_n [1 - \mathbb{E}_{\mathbb{Q}\{u_{ni}\}}\{u_{ni}\}] + K_i - k_i - 1). \end{aligned}$$

Comparing this to the exponential form of the Beta distribution, we see that $\mathbb{Q}\{r_i\}$ is a Beta distribution with parameters

$$\begin{aligned} \hat{\beta}_i^1 &= \beta_i^1 + \sum_n \mathbb{E}_{\mathbb{Q}\{u_{ni}\}}\{u_{ni}\} + k_i \\ \hat{\beta}_i^0 &= \beta_i^0 + \sum_n [1 - \mathbb{E}_{\mathbb{Q}\{u_{ni}\}}\{u_{ni}\}] + K_i - k_i. \end{aligned} \tag{9}$$

The sum is only over coefficients with a new observation.

C. Observation Reliability

Again, by integrating out all the variables except u_{ni}, we have

$$\begin{aligned} \ln(\mathbb{Q}\{u_{ni}\}) = const &+ \mathbb{E}_{\mathbb{Q}\{r_i\}}\{\ln(\mathbb{P}\{u_{ni}|r_i\})\} \\ &+ \mathbb{E}_{\mathbb{Q}\{\phi_n\}}\{\ln(\mathbb{P}\{o_{ni}|u_{ni},\phi_n\})\}. \end{aligned}$$

By employing (5) and (7), we have

$$\begin{aligned} \ln(\mathbb{Q}\{u_{ni}\}) &= u_{ni}\mathbb{E}_{\mathbb{Q}\{r_i\}}\{\ln(r_i)\} + (1-u_{ni})\mathbb{E}_{\mathbb{Q}\{r_i\}}\{\ln(1-r_i)\} \\ &\quad + (1-u_{ni})\ln(0.5) + u_{ni}[o_{ni}\mathbb{E}_{\mathbb{Q}\{\phi_n\}}\{\ln(\phi_n)\} \\ &\quad + (1-o_{ni})\mathbb{E}_{\mathbb{Q}\{\phi_n\}}\{\ln(1-\phi_n)\}] + const. \end{aligned}$$

This update rule, like the others, is a simple expression, as the observations are either 0 or 1. The inference algorithm cannot update u_{ni} if processor i has not reported a measurement on coefficient n. Thus, the update rule is employed for each coefficient on which processor i has a new observation.

To update the distribution, we evaluate the expression for $u_{ni} = 0$ and $u_{ni} = 1$. Since $\mathbb{Q}\{\phi_n\}$ and $\mathbb{Q}\{r_i\}$ are Beta distributions, $\mathbb{E}_{\mathbb{Q}\{\phi_n\}}\{\ln(\phi_n)\}$, $\mathbb{E}_{\mathbb{Q}\{\phi_n\}}\{\ln(1-\phi_n)\}$, $\mathbb{E}_{\mathbb{Q}\{r_i\}}\{\ln(r_i)\}$, and $\mathbb{E}_{\mathbb{Q}\{r_i\}}\{\ln(1-r_i)\}$ can be calculated using (7).

After normalizing the probabilities to have a valid Bernoulli distribution, the parameter of the distribution can be updated as $\tau_{ni} = \mathbb{E}_{\mathbb{Q}\{u_{ni}\}}\{u_{ni}\} = \mathbb{Q}\{u_{ni} = 1\}$.

REFERENCES

[1] D. L. D. Donoho, "Compressed sensing," vol. 52, pp. 1289–1306, 4 2006.

[2] E. E. J. Candès, "Compressive sampling," 2006.

[3] B. Shahrasbi and N. Rahnavard, "Model-Based Nonuniform Compressive Sampling and Recovery of Natural Images Utilizing a Wavelet-Domain Universal Hidden Markov Model," *IEEE Transactions on Signal Processing*, 2017.

[4] D. Needell and R. Ward, "Stable Image Reconstruction Using Total Variation Minimization," *SIAM Journal on Imaging Sciences*, vol. 6, pp. 1035–1058, 1 2013.

[5] J. Tan, Y. Ma, H. Rueda, D. Baron, and G. R. Arce, "Compressive Hyperspectral Imaging via Approximate Message Passing," *IEEE Journal of Selected Topics in Signal Processing*, vol. 10, pp. 389–401, 3 2016.

[6] C. Jeon, R. Ghods, A. Maleki, and C. Studer, "Optimality of large MIMO detection via approximate message passing," in *2015 IEEE International Symposium on Information Theory (ISIT)*, pp. 1227–1231, IEEE, 6 2015.

[7] M. Joneidi, A. Zaeemzadeh, B. Shahrasbi, G.-J. Qi, and N. Rahnavard, "E-optimal Sensor Selection for Compressive Sensing-based Purposes," *IEEE Transactions on Big Data*, p. To Appear in, 2018.

[8] S. Salehi, M. B. Mashhadi, A. Zaeemzadeh, N. Rahnavard, and R. F. De Mara, "Energy-Aware Adaptive Rate and Resolution Sampling of Spectrally Sparse Signals Leveraging VCMA-MTJ Devices," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, pp. 1–1, 2018.

[9] J. Zhu, R. Pilgrim, and D. Baron, "An overview of multi-processor approximate message passing," in *2017 51st Annual Conference on Information Sciences and Systems (CISS)*, pp. 1–6, IEEE, 3 2017.

[10] S. Patterson, Y. C. Eldar, and I. Keidar, "Distributed Compressed Sensing for Static and Time-Varying Networks," *IEEE Transactions on Signal Processing*, vol. 62, pp. 4931–4946, 10 2014.

[11] C. Ravazzi, S. M. Fosson, and E. Magli, "Distributed iterative thresholding for ℓ_0/ℓ_1-regularized linear inverse problems," *IEEE Transactions on Information Theory*, vol. 61, pp. 2081–2100, April 2015.

[12] Y. Ma, Y. M. Lu, and D. Baron, "Multiprocessor approximate message passing with column-wise partitioning," in *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings*, 2017.

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

[14] P. Han, J. Zhu, R. Niu, and D. Baron, "Multi-processor approximate message passing using lossy compression," in *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings*, 2016.

[15] D. Needell and T. Woolf, "An asynchronous parallel approach to sparse recovery," in *2017 Information Theory and Applications Workshop (ITA)*, pp. 1–5, IEEE, 2 2017.

[16] T. Blumensath and M. E. Davies, "Iterative hard thresholding for compressed sensing," *Applied and Computational Harmonic Analysis*, vol. 27, pp. 265–274, 11 2009.

[17] N. Nguyen, D. Needell, and T. Woolf, "Linear Convergence of Stochastic Iterative Greedy Algorithms With Sparse Constraints," *IEEE Transactions on Information Theory*, vol. 63, pp. 6869–6895, 11 2017.

[18] M. I. Jordan, Z. Ghahramani, T. S. Jaakkola, and L. K. Saul, "An introduction to variational methods for graphical models," *Machine learning*, vol. 37, no. 2, pp. 183–233, 1999.

[19] M. J. Wainwright and M. I. Jordan, "Graphical models, exponential families, and variational inference," *Foundations and Trends{®} in Machine Learning*, vol. 1, no. 1-2, pp. 1–305, 2008.

[20] C. M. Bishop, *Pattern recognition and machine learning*. springer, 2006.

[21] L. Dagum and R. Menon, "OpenMP: an industry standard API for shared-memory programming," *IEEE Computational Science and Engineering*, vol. 5, no. 1, pp. 46–55, 1998.

[22] D. L. Donoho, A. Maleki, and A. Montanari, "Message-passing algorithms for compressed sensing," *Proceedings of the National Academy of Sciences*, vol. 106, no. 45, pp. 18914–18919, 2009.