A Coding Theory Approach to Noisy Compressive Sensing Using Low Density Frames

Mehmet Akçakaya, Jinsoo Park, and Vahid Tarokh

Abstract—**We consider the compressive sensing of a sparse or compressible signal $\mathbf{x} \in \mathbb{R}^M$. We explicitly construct a class of measurement matrices inspired by coding theory, referred to as low density frames, and develop decoding algorithms that produce an accurate estimate $\hat{\mathbf{x}}$ even in the presence of additive noise. Low density frames are sparse matrices and have small storage requirements. Our decoding algorithms can be implemented in $O(Md_v^2)$ complexity, where d_v is the left degree of the underlying bipartite graph. Simulation results are provided, demonstrating that our approach outperforms state-of-the-art recovery algorithms for numerous cases of interest. In particular, for Gaussian sparse signals and Gaussian noise, we are within 2-dB range of the theoretical lower bound in most cases.**

Index Terms—**Belief propagation, coding theory, compressive sensing, EM algorithm, Gaussian scale mixtures, low density frames, sum product algorithm.**

I. Introduction

A. Background

LET $\mathbf{x} = (x_1, \ldots, x_M) \in \mathbb{R}^M$ with $\|\mathbf{x}\|_0 = |\{x_i : x_i \neq 0\}| = L$. $\mathbf{x}$ is said to be sparse if $L \ll M$. Consider

$$\mathbf{y} = \mathbf{A}\mathbf{x} + \mathbf{n} \tag{1}$$

where $\mathbf{A}$ is an $N \times M$ measurement matrix and $\mathbf{n} \in \mathbb{R}^N$ is a noise vector. When $N < M$, $\mathbf{y}$ is called the compressively sensed version of $\mathbf{x}$ with measurement matrix $\mathbf{A}$ [14], [20]. In this paper, we are interested in estimating a sparse vector $\mathbf{x}$ from the observed vector $\mathbf{y}$ and the measurement matrix $\mathbf{A}$.

We refer to the case $\mathbf{n} = \mathbf{0}$ as *noiseless compressive sensing*. This is the only case when $\mathbf{x}$ can be perfectly recovered. In particular, it can be shown [15] that if $\mathbf{A}$ has the property that every of its N columns are linearly independent, then a decoder can recover $\mathbf{x}$ uniquely from $N = 2L$ samples by solving the ℓ_0 minimization problem

$$\min \|\mathbf{x}\|_0 \quad \text{s.t.} \quad \mathbf{y} = \mathbf{A}\mathbf{x}. \tag{2}$$

However, solving this ℓ_0 minimization problem for general $\mathbf{A}$ is NP-hard [20]. An alternative solution method proposed in the literature is the ℓ_1 regularization approach, where

$$\min \|\mathbf{x}\|_1 \quad \text{s.t.} \quad \mathbf{y} = \mathbf{A}\mathbf{x} \tag{3}$$

is solved instead. Criteria have been established in the literature as to when the solution of (3) coincides with that of (2) in the noiseless case for various classes of measurement matrices [15], [20]. For $\mathbf{A}$ chosen from a Gaussian ensemble, exact recovery is possible if $N > C_{\ell_1} L \log\left(\frac{M}{L}\right)$ as long as the observations are not contaminated with (additive) noise [15].

It can be shown that there is a relationship between the solution to problem (1) and minimum Hamming distance problem in coding theory [1], [2], [41]. This approach was further exploited in [53]. Using this connection, we constructed ensembles of measurement matrices[1] and associated decoding algorithms that solves the ℓ_0 minimization problem with complexity $O(MN)$ when $N \geq 2L$ [1], [2].

For problem (1) with nonzero $\mathbf{n}$, referred to as *noisy compressive sensing*, the ℓ_1 regularization approach of (3) can also be applied, with an inequality constraint, $\|\mathbf{A}\mathbf{x} - \mathbf{y}\|_2 \leq \epsilon$ (with ϵ related to $\|\mathbf{n}\|_2$). For a measurement matrix $\mathbf{A}$ that satisfies the restricted isometry principle (RIP), solving a quadratic program generates $\hat{\mathbf{x}}_{\mathrm{QP}}$ with $\|\hat{\mathbf{x}}_{\mathrm{QP}} - \mathbf{x}\|_2 \leq C_1\epsilon$, where C_1 is a constant that depends on $\mathbf{A}$ [14]. This method may not be implemented exactly in real time with the limitations of today's hardware. To improve the running time of ℓ_1 methods, the use of sparse matrices for $\mathbf{A}$ have been investigated [8]. Using expansion properties of the graphs represented by such matrices, one obtains $\hat{\mathbf{x}}_{\mathrm{E}}$ with $\|\hat{\mathbf{x}}_{\mathrm{E}} - \mathbf{x}\|_1 \leq C_3\|\mathbf{n}\|_1$, where C_3 is a constant depending on $\mathbf{A}$ [8].

Variants of the matching pursuit algorithm, e.g., subspace pursuit [19] and CoSaMP [35], have also been proposed with similar guarantees to ℓ_1 methods, and complexity $O(\mathcal{L}\log R)$, where $\mathcal{L}$ is the complexity of matrix-vector multiplication and R is a precision parameter bounded above by $\|\mathbf{x}\|_2$ (which is $O(N)$ for a fixed signal-to-noise ratio). An important variant of the matching pursuit algorithm, namely sparse matching pursuit (SMP) is proposed in [9]. This algorithm has $O\left(M\log\left(\frac{M}{L}\right)\right)$ complexity, and for $N > C_{\mathrm{SMP}} L \log\left(\frac{M}{L}\right)$ measurements, outputs $\hat{\mathbf{x}}_{\mathrm{SMP}}$ with $\|\hat{\mathbf{x}}_{\mathrm{SMP}} - \mathbf{x}\|_1 \leq C_4\|\mathbf{n}\|_1$, where C_4 is a constant depending on the sparse matrix $\mathbf{A}$.

Yet another direction in compressive sensing is the use of the Bayesian approach. In [28], the idea of relevance vector machine (RVM) [46] has been applied to compressive sensing. Although simulation results indicate that the algorithm has good performance, it has complexity $O(MN^2)$.

Manuscript received November 16, 2010; revised February 25, 2011; accepted July 19, 2011. Date of publication August 01, 2011; date of current version October 12, 2011. The associate editor coordinating the review of this manuscript and approving it for publication was Prof. Emmanuel Candes.

M. Akçakaya is with the Beth Israel Deaconess Medical Center, Harvard Medical School, Boston, MA 02215 USA (e-mail: akcakaya@seas.harvard.edu).

J. Park and V. Tarokh are with the School of Engineering and Applied Sciences, Harvard University, Cambridge, MA, 02138 USA (e-mail: park10@fas.harvard.edu, vahid@seas.harvard.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TSP.2011.2163402

[1]We use the terms "frame" and "measurement matrix" interchangeably throughout the rest of the paper.

Another work that employs the Bayesian approach using sparse matrices is proposed in [7] and [42]. Similar to our approach, this excellent work uses a generalization of LDPC codes [30], although their recovery technique is different from our proposed method. The current paper improves both on the performance and the complexity of the algorithm presented in [7] and [42]. Further discussions are included in Section III. For completeness, we also note the recent work on belief-propagation-based algorithms for compressive sensing using dense measurement matrices [21], [33].

B. Contributions

In this paper, we study the construction of measurement matrices that can be stored and manipulated efficiently in high dimensions, and fast decoding algorithms that generate estimates with small ℓ_2 distortion. The ensemble of measurement matrices is a generalization of LDPC codes and we refer to them as low density frames (LDFs). For our decoding algorithms, we combine various ideas from coding theory, statistical learning theory and theory of estimation, inspired by the excellent papers of [7], [9], [27], and [28].

More specifically, we combine the sum-product algorithm [29], [30], Gaussian scale mixture distributions [4], [12], [46], the expectation-maximization (EM) algorithm [22], [34] and a pruning strategy [19], in a nontrivial way to design an efficient and high-performance recovery algorithm, which we call the sum product with expectation maximization (SuPrEM) algorithm. We note that although these components have been used in compressive sensing before in [7], [9], [19], [28], [35], and [42], it is the way we put them together that makes SuPrEM algorithm efficient and suitable for sparse recovery for high values of L. With Jeffreys' prior [23], [37], the SuPrEM algorithm does not require any parameters to be specified except the sparsity level L, the noise level and the number of iterations for the message-passing schedule. Simulation results are provided in Section IV indicating an excellent distortion performance at high levels of sparsity and for high levels of noise, as noted in [25]. Furthermore SuPrEM algorithm can be implemented in $O(Md_v^2)$ complexity for a fixed number of iterations, and the exact number of operations required per iteration is specified in Section III-B in terms of the design parameters of the measurement matrix.

The outline of this paper is given next. In Section II, we introduce low density frames and study their basic properties. In Section III, we introduce various concepts used in our algorithm and describe the decoding algorithm. In Section IV, we provide extensive simulation results for a number of different testing criteria. Finally in Section V, we make our conclusions and provide directions for future research.

II. Low Density Frames

We define a (d_v, d_c)-regular low density frame (LDF) as a $N \times M (N < M)$ matrix $\mathbf{F}$ that has d_c nonzero elements in each row and d_v nonzero elements in each column, and such that $\frac{(Md_v)}{(MN)} \ll 1$. Clearly $Md_v = Nd_c$. We also note that the redundancy of the frame is $r = \frac{M}{N} = \frac{d_c}{d_v}$. We will restrict ourselves to *binary* regular LDFs, where the nonzero elements of $\mathbf{F}$ are ones.

The main connection between the ℓ_0 minimization problem and coding theory involves the description of the underlying code [1], $\mathcal{V}$ of $\mathbf{F}$, where

$$\mathcal{V} = \{\mathbf{d} \in \mathbb{R}^M : \mathbf{Fd} = \mathbf{0}\}.$$

One can view $\mathcal{V}$ as the set of vectors whose product with each row of $\mathbf{F}$ "checks" to 0. In the works of Tanner, it was noted that this relationship between the checks and the codewords of a code can be represented by a bipartite graph [45]. This bipartite graph consists of two disjoint sets of vertices, V and C, where V contains the variable nodes and C contains the factor nodes representing checks that codewords need to satisfy. Thus, we have $|V| = M$ and $|C| = N$. Furthermore node j in V will be connected to node i in C if and only if the $(i,j)^{\text{th}}$ element of $\mathbf{F}$ is nonzero. Thus, the number of edges of the graph is equal to the number of nonzero elements in the measurement matrix $\mathbf{F}$. For an LDF, this leads to a sparse bipartite graph. We use factor graphs [13] for our representation. For representation of LDFs, it is convenient to use a factor node, depicted by a $\boxplus$, called a parity check node. Specifically, we use the syndrome representation [31], where the variable nodes connected the j^{th} parity check node sum to r_j. Thus, the graph now represents the set $\{\mathbf{d} \in \mathbb{R}^M : \mathbf{Fd} = \mathbf{r}\}$ which is a coset of the underlying code of the frame.

The natural inference algorithm for factor graphs is the sum-product algorithm [13]. This algorithm is an exact inference algorithm for tree-structured graphs (i.e., graphs with no cycles), and is usually described over discrete alphabets. However, the ideas also apply to continuous random variables with the sum being replaced by integration. In doing so, the computational cost of implementation increases and this issue will be addressed later. It is important to note that the graph representing an LDF will have cycles. Without the tree structure, the sum-product algorithm will only be an approximate inference algorithm. However, in coding theory, it has been empirically shown that for sparse graphs this approximate algorithm works very well, achieving the capacity of Gaussian channels within a fraction of a decibel [30], [39].

III. Sum Product Algorithm With Expectation Maximization

Given a set of observations, the sum-product algorithm (SPA) can be used to approximate the posterior marginal distributions [29], [30]. In fact, when there is no noise, variants of this algorithm [44] has been successfully adapted to compressive sensing [41], [53]. However, for the practical case of noisy observations, these algorithms no longer can be applied in a straightforward manner, because the random variables are continuous. Some authors have tried to circumvent this difficulty by using a two-point Gaussian mixture approach [7], [42]. In their work, each component x_i is modeled with the probability density $p_{x_i}(x_i) = \frac{L}{M} \cdot \mathcal{N}(x_i|0, \sigma_1^2) + \left(1 - \frac{L}{M}\right) \cdot \mathcal{N}(x_i|0, \sigma_0^2)$, where $\mathcal{N}(x|a, A)$ denotes the Gaussian probability density function with mean a and variance A. σ_1 is usually chosen to be a constant multiple (e.g., 10) times bigger than σ_0. The component with variance σ_1^2 models the nonzero coefficients of $\mathbf{x}$, whereas the other component models the near-zero coefficients of $\mathbf{x}$. For recovery, the sum-product algorithm is used with this probability distribution on x_i. The complexity of this algorithm may grow quickly as the number of Gaussian components in the mixtures could grow exponentially, unless some approximation is applied. On the other hand, using these approximations degrades the performance of the LDF approach. Another approach is to use sampling from a product of distributions [11] which is also used in the implementation of [7]. This results in a large constant overhead for the algorithm, although the complexity is $O(M\log^2(M))$ [7]. Furthermore, the accuracy of the algorithm also depends on the values of σ_1 and σ_0 which need to be chosen properly based on knowledge of $\|\mathbf{x}\|_2$.

In this paper, we consider Gaussian scale mixture (GSM) priors with Jeffreys' noninformative hyperprior to devise an estimation algorithm for the noisy compressive sensing problem

$$\mathbf{r} = \mathbf{F}\mathbf{x} + \mathbf{n},$$

as well as the noiseless problem. Throughout the paper, we assume that

$$\mathbf{n} \sim \mathcal{N}(0, \sigma^2\mathbf{I}_N).$$

However, simulation results (not included in this paper) indicate that the algorithms still work well even for non-Gaussian noise. We define the signal-to-noise ratio (SNR) as

$$\mathrm{SNR} = 10\log_{10}\frac{\|\mathbf{F}\mathbf{x}\|^2}{\mathbb{E}_{\mathbf{n}}\|\mathbf{n}\|^2} = 10\log_{10}\frac{\|\mathbf{F}\mathbf{x}\|^2}{\sigma^2 N}.$$

A. Gaussian Scale Mixtures

The main difficulty in using SPA in compressive sensing setting is that the variables of interest are continuous. Nonetheless SPA can be employed when the underlying continuous random variables are Gaussian [50]. Since any Gaussian pdf $\mathcal{N}(x|a, A)$ can be determined by its mean a and variance A, these constitute the messages in this setting. At the variable nodes, the product of Gaussian probability density functions (pdf) will result in a (scaled) Gaussian pdf, and at the check nodes, the convolution of Gaussian pdfs will also result in a Gaussian pdf, i.e.,

$$\mathcal{N}(x|a_1, A_1) * \mathcal{N}(x|a_2, A_2) \propto \mathcal{N}(x|a_1 + a_2, A_1 + A_2)$$

and

$$\mathcal{N}(x|a_1, A_1) \cdot \mathcal{N}(x|a_2, A_2) \propto \mathcal{N}(x|b, B)$$

where $\propto$ denotes normalization up to a constant, $B = (A_1^{-1} + A_2^{-1})^{-1}$, and $b = B(A_1^{-1}a_1 + A_2^{-1}a_2)$. We note that all the underlying operations for SPA preserve the Gaussian structure.

However, the Gaussian pdf is not "sparsity-enhancing."[2] Thus, some authors propose the use of the Laplace prior, $p(\mathbf{x}) = \prod_i p_{x_i}(x_i) = \prod_i \frac{\lambda}{2}\exp(-\lambda|x_i|)$. Clearly, with this prior and for Gaussian noise $\mathbf{n}$

$$p(\mathbf{x}|\mathbf{y}) \propto p(\mathbf{y}|\mathbf{x})p(\mathbf{x}) \propto \exp\left(-\|\mathbf{y} - \mathbf{A}\mathbf{x}\|_2^2 - \lambda'\|\mathbf{x}\|_1\right) \quad (4)$$

and maximization of $p(\mathbf{x}|\mathbf{y})$ is equivalent to minimizing $\|\mathbf{y} - \mathbf{A}\mathbf{x}\|_2^2 + \lambda'\|\mathbf{x}\|_1$, which is the objective function for the LASSO algorithm [49]. However, a straightforward minimization of this function or an appropriate *a priori* choice of λ may not be feasible. We also note that there is a strand of work [5], [22], [24] on developing lower complexity algorithms which find solutions that approximately minimize this objective function.

[2]By a sparsity enhancing (generalized) distribution, we mean that the maximum *a posteriori* (MAP) estimator of $\mathbf{x}$, which outputs $\arg\max p(\mathbf{x}|\mathbf{y})$ favors sparse vectors in the presence of Gaussian noise with mean 0 and covariance matrix $\sigma^2\mathbf{I}_N$. We note that the MAP estimate is given by $\arg\max \exp\left(-\frac{1}{2\sigma^2}\|\mathbf{y} - \mathbf{A}\mathbf{x}\|_2^2\right)p(\mathbf{x})$. When $p(\mathbf{x})$ is a multivariate Gaussian with independent components, this amounts to the minimization of the ℓ_2 norm of $\mathbf{x}$ subject to a distance constraint on $\|\mathbf{y} - \mathbf{A}\mathbf{x}\|_2$, which, in general, does not favor sparse vectors.

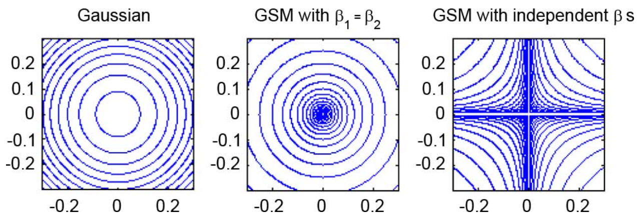


Fig. 1. Contour plots for a Gaussian distribution (left), a GSM with $\beta_1 = \beta_2$ distributed according to Jeffreys' prior (middle), a GSM with β_1 and β_2 i.i.d. with Jeffreys' prior (right).

In this paper, we consider the family of GSM densities [37], given by $x = \sqrt{\beta}u$, where u is a zero-mean Gaussian and β is a positive scalar random variable. This family of densities has been successfully used in image processing [37], and learning theory [13]. In our model, we use Jeffreys' prior for $p_\beta(\beta)$, which in this case is given by $p_{\beta_i}(\beta_i) = \frac{1}{\beta_i}$. We note that this is an improper density, i.e., it cannot be normalized. In Bayesian statistics, this kind of improper priors are used frequently, since only the relative weight of the prior determines the *a posteriori* density [40].

As depicted in the middle subplot of Fig. 1, compared to a Gaussian distribution, a GSM with β_i distributed according to Jeffreys' prior has a much sharper peak at the origin even when $\beta_1 = \beta_2$. However, the subplot on the right demonstrates that if the β_is are indeed independent, the GSM will be highly concentrated not only around the origin, but along the coordinate axes as well, which is a desired property if we have no further information about the locations of the sparse coefficients of $\mathbf{x}$ [47]. Thus, in our model, we will assume that $p(\mathbf{x}, \boldsymbol{\beta}) = \prod_{i=1}^M p(x_i|\beta_i)\prod_{i=1}^M p(\beta_i)$ in order to enhance sparsity in all coordinates. This independence assumption is natural and commonly used in the literature [23], [49].

B. SuPrEM Algorithm

The factor graph for decoding purposes is depicted in Fig. 2. Here, $\mathbf{r}$ is the vector of observed variables, $\mathbf{x}$ is the vector of hidden variables and $\boldsymbol{\beta}$ is the vector of parameters. At every iteration t, this algorithm uses a combination of the Sum-Product Algorithm (SPA) and EM algorithm [34] to generate estimates for the hyperpriors $\{\beta_k^{(t)}\}$, as well as a point estimate $\{\hat{x}_k^{(t)}\}$. In the EM stage of the algorithm, $Q(\boldsymbol{\beta}|\boldsymbol{\beta}^{(t)})$ for the **E-step** is given by

$$\begin{aligned}
Q(\boldsymbol{\beta}|\boldsymbol{\beta}^{(t)}) &= \mathbb{E}_{\mathbf{x}}\left(\log p(\mathbf{x}, \mathbf{y}, \boldsymbol{\beta})\middle|\mathbf{y}, \boldsymbol{\beta}^{(t)}\right) \\
&= \mathbb{E}_{\mathbf{x}}\left(\log p(\mathbf{y}|\mathbf{x})\middle|\mathbf{y}, \boldsymbol{\beta}^{(t)}\right) \\
&\quad + \mathbb{E}_{\mathbf{x}}\left(\log p(\mathbf{x}, \boldsymbol{\beta})\middle|\mathbf{y}, \boldsymbol{\beta}^{(t)}\right) \\
&= C_1 + \sum_{i=1}^M \mathbb{E}_{\mathbf{x}}\left(\log p(x_i, \beta_i)\middle|\mathbf{y}, \boldsymbol{\beta}^{(t)}\right) \\
&= C_1 + \sum_{i=1}^M Q(\beta_i|\boldsymbol{\beta}^{(t)}),
\end{aligned}$$

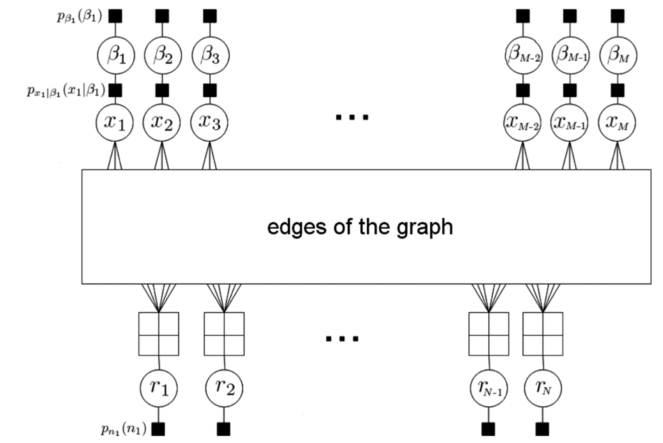


Fig. 2. Factor graph for a (3,6)-regular LDF with the appropriate hyperpriors.

where $C_1 = \mathbb{E}_{\mathbf{x}}\big(\log p(\mathbf{y}|\mathbf{x})\big|\mathbf{y}, \boldsymbol{\beta}^{(t)}\big)$ is a term independent of $\boldsymbol{\beta}$, and $Q(\beta_i|\boldsymbol{\beta}^{(t)}) = \mathbb{E}_{x_i}\big(\log p(x_i, \beta_i)|\mathbf{y}, \boldsymbol{\beta}^{(t)}\big)$.

Since in our setting, the underlying variables are Gaussian, the density $p(x_i|\mathbf{y}, \boldsymbol{\beta}^{(t)})$ produced by the SPA is also Gaussian, with mean μ_i and variance ν_i. One can explicitly write out $Q(\beta_i|\boldsymbol{\beta}^{(t)})$ as

$$\begin{aligned} Q(\beta_i|\boldsymbol{\beta}^{(t)}) &= \mathbb{E}_{x_i}\left(\log p(x_i, \beta_i)\big|\mathbf{y}, \boldsymbol{\beta}^{(t)}\right) \\ &= \mathbb{E}_{x_i}\left(\log\left(\frac{1}{\sqrt{2\pi\beta_i}}\exp\left(-\frac{x_i^2}{2\beta_i}\right)\frac{1}{\beta_i}\right)\right)\Big|\mathbf{y}, \boldsymbol{\beta}^{(t)}\Bigg) \\ &= C_2 - \frac{3}{2}\log\beta_i - \frac{1}{2\beta_i}\mathbb{E}_{x_i}(x_i^2|\mathbf{y}, \boldsymbol{\beta}^{(t)}) \\ &= C_2 - \frac{3}{2}\log\beta_i - \frac{1}{2\beta_i}(\mu_i^2 + \nu_i) \end{aligned} \tag{5}$$

where C_2 is independent of β_i.

For the **M-step**, we find $\boldsymbol{\beta}^{(t+1)} = \arg\max_{\boldsymbol{\beta}} Q(\boldsymbol{\beta}|\boldsymbol{\beta}^{(t)})$. Clearly $Q(\boldsymbol{\beta}|\boldsymbol{\beta}^{(t)})$ can be maximized by maximizing each $Q(\beta_i|\boldsymbol{\beta}^{(t)})$. Hence we have the simple local update rule

$$\beta_i^{(t+1)} = \arg\max_{\beta_i} Q(\beta_i|\boldsymbol{\beta}^{(t)}) = \frac{\mu_i^2 + \nu_i}{3}. \tag{6}$$

We note that although we are using a specific GSM distribution that utilizes Jeffreys' hyperprior, the algorithm can be used with any GSM distribution. The family of GSM distributions include the Laplace prior (frequently used in ℓ_1 regularized algorithms), the generalized Gaussian prior (frequently used in ℓ_p minimization with $p < 1$) and various other heavy-tailed densities [4], [12]. The use of other priors from the GSM family would lead to different update equations in the EM algorithm, which depend on the parameter used in that specific prior (e.g., λ in the Laplace prior). This results in an additional parameter which needs to be fine-tuned. On the other hand, the use of Jeffreys' prior results in a simple update rule that does not depend on any parameters.

At each iteration, the sum-product algorithm can be used in combination with the aforementioned EM update rule. However, this algorithm does not enforce sparsity, thus it is not well-suited for situations where the output needs to be sparse, such as the noiseless compressive sensing setting. To this end, we propose our main algorithm, which is called the SuPrEM Algorithm. An implementation of this algorithm is available in [56]. This algorithm enforces sparsity at various stages of the algorithm and sends messages between the nodes of the underlying graph accordingly. We keep a set of candidate variable nodes $\mathcal{O}$ that are likely to have nonzero values, and modify the messages from the variable nodes that do not belong to a specified subset of $\mathcal{O}$ denoted by $\mathcal{O}_1$ [3]. Similar ideas have been used in developing state-of-the-art recovery algorithms for compressive sensing, such as sparse matching pursuit [9], subspace pursuit [19], and CoSaMP [35]. The full description is given in Algorithm 1.

Algorithm 1: SuPrEM Algorithm

- **Inputs**: The observed vector $\mathbf{r}$, the measurement matrix $\mathbf{F}$, the sparsity level L, the noise level σ^2 (optional), and a message-passing schedule $\mathcal{S}$.

1) **Initialization:** Let $\beta_k^{(0)} = \frac{|(\mathbf{F}^T\mathbf{r})_k|^2}{d_v^2}$ and let $\mathcal{O}_1 = \emptyset$. Initial outgoing messages from variable node x_k is $(0, \beta_k^{(0)})$.
2) **Check Nodes:** For $i = 1, 2, \ldots, N$, let $\{i_1, i_2, \ldots, i_{d_c}\}$ be the indices of the variable nodes connected to the i^{th} check node r_i. Let the message coming from variable node x_{i_j} to the check node r_i at t^{th} iteration be $(\mu_{i_j}^{(t)}, \nu_{i_j}^{(t)})$ for $j = 1, \ldots, d_c$. Then the outgoing message from check node r_i to variable node x_{i_j} is $\left(r_i - \sum_{k=1,k\neq j}^{d_c} \mu_{i_k}^{(t)}, \sum_{k=1,k\neq j}^{d_c} \nu_{i_k}^{(t)} + \sigma^2\right)$. The messages are sent according to the schedule $\mathcal{S}$.
3) **Variable Nodes:** For $k = 1, 2, \ldots, M$, let $\{k_1, k_2, \ldots, k_{d_v}\}$ be the indices of the check nodes connected to the k^{th} variable node x_k. Let the incoming message from the check node r_{k_j} to the variable node x_k at the t^{th} iteration be $(\mu_{k_j}^{(t)}, \nu_{k_j}^{(t)})$ for $j = 1, \ldots, d_v$.
   a. *EM update:* Let $V_k^{(t)} = \left(\sum_{j=1}^{d_v} \frac{1}{\nu_{k_j}^{(t)}} + \frac{1}{\beta_k^{(t-1)}}\right)^{-1}$ and $\mu_k^{(t)} = V_k^{(t)}\left(\sum_{j=1}^{d_v} \frac{\mu_{k_j}^{(t)}}{\nu_{k_j}^{(t)}}\right)$. Then the EM update is $\beta_k^{(t)} = \frac{(\mu_k^{(t)})^2 + V_k^{(t)}}{3}$.
   b. *Message updates:* The outgoing message from variable node x_k to check node r_{k_i} at the $(t+1)^{\text{th}}$ iteration is given by $(\mu_{k_i}^{(t+1)}, \nu_{k_i}^{(t+1)})$, where $\nu_{k_i}^{(t+1)} = \left(\sum_{j=1,j\neq i}^{d_v} \frac{1}{\nu_{k_j}^{(t)}} + \frac{1}{\beta_k^{(t)}}\right)^{-1}$ and $\mu_{k_i}^{(t+1)} = \nu_{k_i}^{(t+1)}\left(\sum_{j=1,j\neq i}^{d_v} \frac{\mu_{k_j}^{(t)}}{\nu_{k_j}^{(t)}}\right)$. The messages are sent according to the schedule $\mathcal{S}$. Also make decisions on $\{\hat{x}_k^{(t)}\} = \{\mu_k^{(t)}\}$.
4) **Sparsification:**
   a. Let $\mathcal{O}_2$ be the indices of the L largest β_ks.
   b. Merge $\mathcal{O}_1$ and $\mathcal{O}_2$, i.e., Let $\mathcal{O} = \mathcal{O}_1 \cup \mathcal{O}_2$.
   c. Identify the indices corresponding to the L largest (in absolute value) coefficients of $\{\hat{x}_k\}_{k\in\mathcal{O}}$.
   d. The variable vertices $k \in \mathcal{O}_1$, send out their messages as was decided in Step 3. The variable vertices $k \notin \mathcal{O}_1$, send out their messages with 0 mean and the variance that was decided in Step 3. Also set $\hat{x}_k = 0$ for $k \notin \mathcal{O}_1$.

5) **Decisions:** Make decisions only the vertices in $\mathcal{O}$. Once these are calculated, keep the L indices with the largest $|\hat{x}_k|, k \in \mathcal{O}$. Set all other indices to 0.
6) **Iterations:** Repeat **(2)–(5)** until convergence.

- **Output:** The estimate is $\hat{\mathbf{x}} = (\hat{x}_1, \hat{x}_2, \dots, \hat{x}_M)^T$.

The main modification is the addition of a sparsification step. Intuitively $\beta_k^{(t)}$ is the reliability of the hypothesis $\hat{x}_k^{(t)} \neq 0$. Throughout the algorithm we maintain a list of variable nodes $\mathcal{O}_1$ that correspond to the largest L coefficients of $\hat{\mathbf{x}}^{(t)}$ at iteration t. We also keep a list of variable nodes $\mathcal{O}_2$ corresponding to the L largest elements of $\boldsymbol{\beta}^{(t)}$, i.e., those with the largest reliabilities of the hypothesis $\hat{x}_k^{(t)} \neq 0$. In the sparsification stage, these two sets are merged, $\mathcal{O} = \mathcal{O}_1 \cup \mathcal{O}_2$. The addition and deletion of elements from $\mathcal{O}$ allow refinements to be made with each iteration. We note $L \leq |\mathcal{O}| \leq 2L$ at any given iteration. Decisions are made on the elements of $\mathcal{O}$, and $\mathcal{O}_1$ is updated. Finally for variable nodes not in $\mathcal{O}_1$, the mean value of the messages is forced to be 0, but the variance (i.e., the uncertainty about the estimate itself) is kept. By modifying the messages this way, we not only enforce sparsity at the final stage, but also throughout the algorithm.

The inputs to the algorithm contain a message-passing schedule $\mathcal{S}$, which turns out to be important for achieving the maximum performance. To this end, we develop a message-passing schedule that attains such good performance and we describe this schedule in detail in the Appendix. For all our simulations, we use this fixed schedule. Simulation results indicate that with this fixed schedule, the algorithm is robust in various different scenarios. We note that the noise level σ^2 is an optional input to the algorithm. Our simulations indicate that the algorithm works without this knowledge also. However, if this extra statistical information is available, it is easily incorporated into the algorithm and results in a performance increase.

SuPrEM algorithm can be implemented with $O(Md_v^2)$ complexity for a fixed number of iterations. For a simple flooding schedule, the message passing and EM part take $M(d_v(2d_c + 3) - 3)$ additions, $M(d_v^2 + 2d_v + 2)$ divisions, $M(2d_v + 3)$ multiplications are required per iteration. Also Md_v additions and $2M$ multiplications are required for the initialization step. The determination of the largest L elements of $\boldsymbol{\beta}$ and $\hat{\mathbf{x}}$. This could be done with $O(M)$ complexity, as described in [18] (Chapter 9). A more straightforward implementation (used in [56]) for this stage uses sorting of the relevant coefficients, which would result in a higher complexity of $O(M \log M)$ for the overall algorithm. The total complexity of the algorithm per iteration is given by $M(d_v(2d_c + 3) - 3)$ additions, $M(d_v^2 + 2d_v + 2)$ divisions, $M(2d_v + 3)$ multiplications and 2 thresholding operations for identifying the L largest components of two vectors. We note that in our simulations $d_v = 3$ and d_c changes based on M and N. The exact number of operations may change based on the message-passing schedule.

C. Reweighted Algorithm

Simulation results show that the SuPrEM algorithm produces estimates with low distortion even for high $\frac{L}{N}$ ratios. However, more iterations are needed to achieve very low distortion levels, which may be undesirable. Thus, we propose a modification to SuPrEM to speed up the convergence that uses estimates generated within a few iterations. In compressive sensing, employing prior estimates to improve the final solution has been used in the iteratively reweighted ℓ_1 (IRL1) technique [16], but this increases the running time by a factor of reweighing steps.

Next, we motivate for our reweighing approach. In our algorithms, the initial choice of $\left\{\beta_k^{(0)} = \frac{|(\mathbf{F}^T\mathbf{r})_k|^2}{d_v^2}\right\}$ is based on the intuition that β_k must be proportional to $|x_k|^2$. By providing a better estimate for the initial $\{\beta_k^{(0)}\}$, the rate of convergence may be improved. The algorithm is initiated with $\boldsymbol{\beta}^{(0)}$ as above and is run for T_{r_1} iterations. At the end of this stage, we reinitialize $\boldsymbol{\beta}^{(0)'}$ to be

$$\beta_k^{(0)'} = \left|\hat{x}_k^{(T_{r_1})}\right|^2 + \frac{\left|(\mathbf{F}^T(\mathbf{r} - \mathbf{F}\hat{\mathbf{x}}^{(T_{r_1})}))_k\right|^2}{d_v^2},$$

and the algorithm is run for T_{r_2} iterations. This process is repeated recursively until convergence or $\mathcal{R}$ times. We note that $\sum_{k=1}^{\mathcal{R}} T_{r_k} = T$, where T is the original number of fixed iterations. Thus, the total number of iterations remains unchanged when we use reweighing.

IV. Simulation Details

A. Simulation Setup

In our simulations, we used LDFs with parameters (3,6), (3,12) and (3, 24) for $\frac{M}{N} = 2, 4, 8$ and $M = 10\,000$. We constructed these frames using the progressive edge growth algorithm [26], avoiding cycles of length 4 when possible.[3] Simulations will be presented for SNR = 12, 24, 36 dB, as well as the noiseless case. For various choices of L and SNR, we ran 1000 Monte Carlo simulations for each value, where $\mathbf{x}$ is generated as a signal with L nonzero elements that are picked from a Gaussian distribution. The support of $\mathbf{x}$ is picked uniformly at random. Once $\mathbf{x}$ is generated, it is normalized such that $\|\mathbf{F}\mathbf{x}\|_2 = \sqrt{N}$. Thus, $\text{SNR} = 10\log_{10}\frac{1}{\sigma^2}$.

Let $\mathcal{G}$ be the genie decoder that has full information about $\text{supp}(\mathbf{x}) = \{i : x_i \neq 0\}$. Let the output of this decoder be $\hat{\mathbf{x}}_{\text{genie}} = \mathcal{G}(\mathbf{r})$ obtained by solving the least squares problem involving $\mathbf{r}$ and the matrix formed by the columns of $\mathbf{F}$ specified by $\text{supp}(\mathbf{x})$. We define the genie distortion measure

$$\bar{d}_g(\mathbf{x}, \hat{\mathbf{x}}_{\text{genie}}) = \frac{\|\mathbf{x} - \hat{\mathbf{x}}_{\text{genie}}\|_2^2}{\|\mathbf{x}\|_2^2}.$$

This distortion measure is invariant to the scaling of $\mathbf{x}$ for a fixed SNR. For any other recovery algorithm that outputs an estimate $\hat{\mathbf{x}}$, we let

$$\bar{d}_e(\mathbf{x}, \hat{\mathbf{x}}_e) = \frac{\|\mathbf{x} - \hat{\mathbf{x}}_e\|_2^2}{\|\mathbf{x}\|_2^2}$$

where the subscript e denotes the estimation procedure. We will be interested in the performance of an estimation procedure with respect to the genie decoder. To this end, we define

$$\mathcal{D}_{\frac{e}{g}}(\mathbf{x}, \hat{\mathbf{x}}_e, \hat{\mathbf{x}}_{\text{genie}}) = \frac{\|\mathbf{x} - \hat{\mathbf{x}}_e\|_2^2}{\|\mathbf{x} - \hat{\mathbf{x}}_{\text{genie}}\|_2^2} = \frac{\bar{d}_e(\mathbf{x}, \hat{\mathbf{x}}_e)}{\bar{d}_g(\mathbf{x}, \hat{\mathbf{x}}_{\text{genie}})}.$$

[3]We also tested LDFs with 4 cycles and this does not seem to have an adverse effect on the average distortion in the presence of noise.

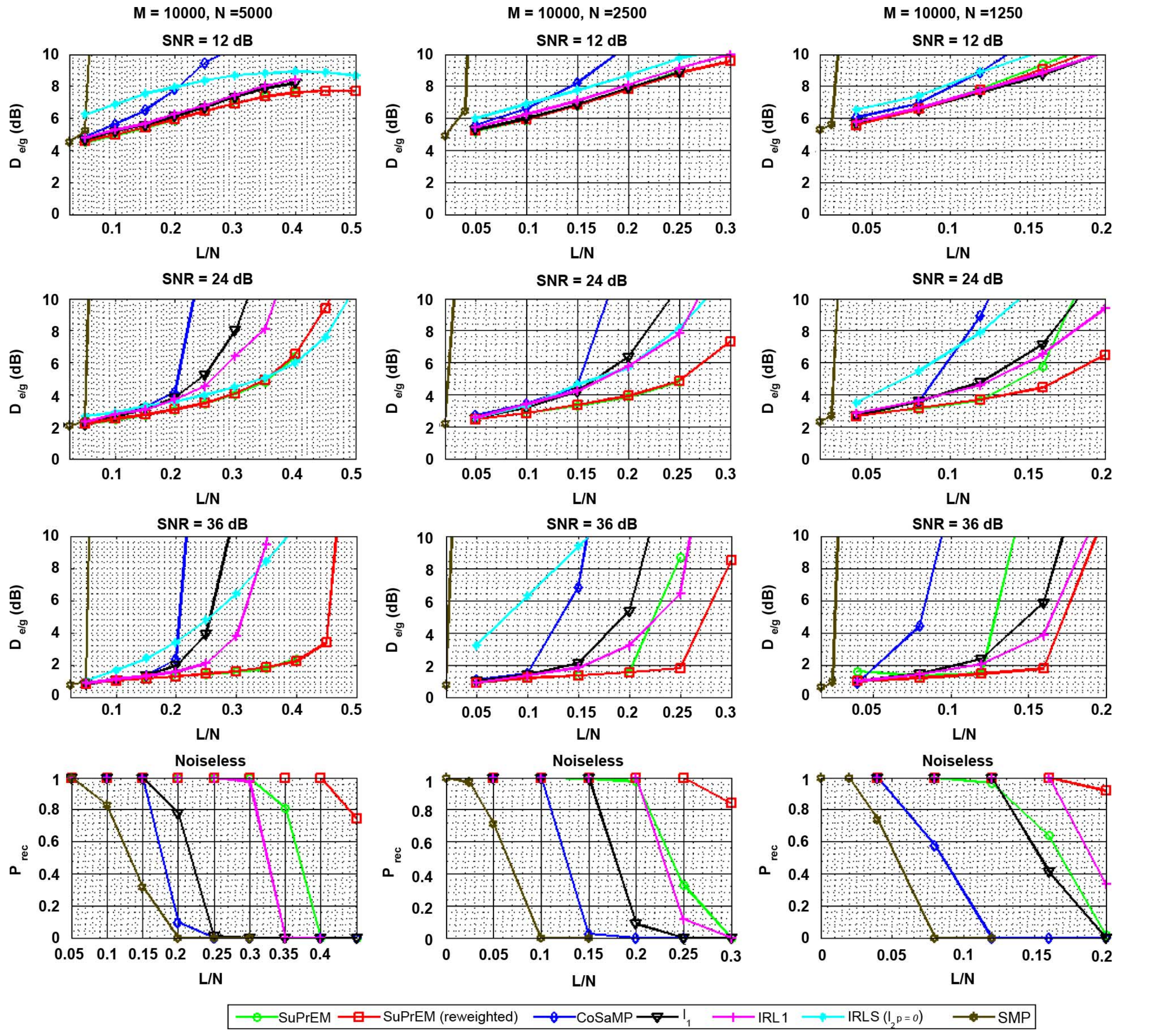


Fig. 3. Performance comparison of recovery algorithms for sparse signals with Gaussian nonzero components.

We will be interested in this quantity averaged over K Monte-Carlo simulations, and converted to decibels. The closer this quantity is to 0 dB means the closer the performance of the estimation procedure is to the performance of the genie decoder.

In other cases, such as the noiseless case, we will be interested in the empirical probability of recovery. For K Monte-Carlo simulations, this is given by

$$P_{\mathrm{rec}} = \frac{1}{K}\sum_{k=1}^{K} \mathbb{I}(\mathbf{x} \sim \hat{\mathbf{x}}_e)$$

where $\mathbb{I}(\cdot)$ is the indicator function for $(\cdot)$ (1 if $(\cdot)$ is true, 0 otherwise). We will define the relation $\mathbf{x} \sim \hat{\mathbf{x}}_e$ to be true only if $\mathrm{supp}(\mathbf{x}) = \mathrm{supp}(\hat{\mathbf{x}}_e)$, unless otherwise specified.

The convergence criterion we used was the convergence of $\{\|\mathbf{r} - \mathbf{F}\hat{\mathbf{x}}^{(t)}\|_2\}_t$. The maximum number of iterations was set to be $T = 500$, chosen to make sure that the algorithms did not stop prematurely. The message passing schedule $\mathcal{S}$ is described in detail in the Appendix. Finally, for the reweighted algorithm we use ten reweighings with $T_{r_1} = \cdots = T_{r_{10}} = \frac{T}{10}$.

B. Simulation Results

Simulation results are presented in Fig. 3 for exactly sparse signals, whose nonzero elements are chosen independently from a Gaussian distribution.

For comparison to our algorithms, we include results for CoSaMP [35] and ℓ_1 based methods [14]–[16], [20], [24]. For these algorithms we used partial Fourier matrices as measurement matrices. The choice of these matrices is based on their small storage requirements (in comparison to Gaussian matrices), while still satisfying restricted isometry principles. For CoSaMP, we used 100 iterations of the algorithm (and 150 iterations of Richardson's iteration for calculating least squares solutions). For ℓ_1 based methods, we used the `L1MAGIC` package in the noiseless case. In the noisy case, we used both `L1MAGIC`, and the `GPSR` package (with Barzilai–Borwein gradient projection with continuation and debiasing). Since these two methods approximately perform the same, we include the results for GPSR here. In the implementation of GPSR we fine-tune the value of τ and observe that $\tau = 0.001\|\mathbf{F}^T\mathbf{r}\|_\infty$ gives the best performance. For the reweighted ℓ_1 approach we used the SPGL1 package [48]. The use of four reweightings as suggested in [16] did not result in a drastic performance increase, however we found that 15 reweightings improved the performance with respect to ℓ_1 minimization. For each set of weights, the algorithm was run for a maximum of 500 iterations. The stability parameter was chosen as described in [16]. For SMP, we used the SMP package [9]. We used a matrix with left degree 50 generated by this package, 500 iterations and convergence factor of 0.1. The left degree and the convergence factor were fine-tuned by hand for highest performance. We note that an improved version of SMP, called sequential SMP (SSMP) may provide better performance for these simulations [10]. However, this algorithm is not included in the comparisons, since its code was not available online at the time of the preparation of this manuscript.

Finally, we note that for the noisy case, the sparsity regularizer corresponding to the Jeffreys' prior used in this text is proportional to $\sum_k \log|x_k|$, which is in one-to-one correspondence with ℓ_p norm of $\mathbf{x}$ as $p \to 0$ [51]. Thus, we also compare our methods to iteratively reweighted least squares (IRLS) algorithm that is used as a heuristic for ℓ_p-norm regularized compressive sensing (with $p = 0$), as in [17]. We note the latter does not utilize the sparsity of LDFs in any particular way. We use the ϵ-continuation scheme in the implementation of [17], however we start with $\epsilon = \frac{1}{100}$ instead of $\epsilon = 1$, which was fine-tuned for the scaling of $\mathbf{x}$ used in this paper to allow for convergence within 500 iterations. We emphasize that ℓ_p-norm regularized IRLS reconstruction was used with LDFs to provide a clear comparison of the possible improvements our method (designed for sparse matrices) has over a more general approach that works with any measurement matrix.

Since the outputs of IRLS and ℓ_1 norm based methods are not sparse, we threshold $\mathbf{x}$ to its L largest coefficients and postulate these are the locations of the sparse coefficients. For all methods, we solve the least squares problem involving $\mathbf{r}$ and the matrix formed by the columns of $\mathbf{F}$ specified by the locations of the sparse coefficients from the final estimate. For partial Fourier matrices we use Richardson's iteration to calculate this vector, whereas for LDFs and the SMP matrix we use the LSQR algorithm which also has $O(M)$ complexity [36].

C. Discussion of the Results

The simulation results indicate that the SuPrEM algorithms outperform the other state-of-the-art algorithms. In the low SNR regime (SNR = 12 dB), SuPrEM algorithms, ℓ_1 based methods and ℓ_p-norm regularized IRLS have similar performance. In high SNR regimes, we see that SuPrEM algorithms outperform the other algorithms both in terms of distortion and in terms of the maximum sparsity they can work at. Also IRL1 technique uniformly outperforms ℓ_1 regularization as expected. For acceleration rate 8, it also performs close to reweighted SuPrEM, although for rates 2 and 4, it is also outperformed in most cases by regular SuPrEM. An interesting point is that the ℓ_p-norm regularized IRLS with LDFs outperforms most other methods that use Fourier matrices for acceleration rate 2. However, for higher acceleration rates, its performance deteriorates, which may be because it is not designed for sparse matrices and thus may be stuck on local minima. We also note that SuPrEM algorithm generally outperforms the IRLS method in a direct comparison (since both have a similar objective function and both are tested with sparse matrices), because the algorithm is specifically designed for LDFs. Furthermore for different values of N, the maximum sparsity scales as $L = O\left(\frac{N}{\log}\left(\frac{M}{N}\right)\right)$, which is the same scaling as those of other methods. We also observe that the reweighted SuPrEM algorithm outperforms the regular SuPrEM algorithm, even though the maximum number of iterations are the same.

Finally, compared to the other methods for the noiseless problem, the SuPrEM algorithms can recover signals that have a higher number of nonzero elements. In this case, the reweighted algorithm performs the best, and converges faster. We also note that the results presented for SMP, CoSaMP and ℓ_1 based methods for the noiseless case are optimistic, since we declare success in recovery if $\bar{d}_e(\mathbf{x}, \hat{\mathbf{x}}_e) < 10^{-6}$. We needed to introduce this measure, since these algorithms tend to miss a small portion of the support of $\mathbf{x}$ containing elements of small magnitude.

We also note that for both partial Fourier matrices and LDFs, the quantity $\bar{d}_g(\mathbf{x}, \hat{\mathbf{x}}_{\text{genie}})$ is almost the same for a fixed L and SNR. This means that $\mathcal{D}_{\frac{e}{g}}(\mathbf{x}, \hat{\mathbf{x}}_e, \hat{\mathbf{x}}_{\text{genie}})$ provides an objective performance criterion in terms of relative mean-square error with respect to the genie bound, as well as in terms of absolute distortion error $\bar{d}_e(\mathbf{x}, \hat{\mathbf{x}}_e)$.

D. Simulation Results for Natural Images

For the testing of compressible signals, instead of using artificially generated signals, we used real-world compressible signals. In particular, we compressively sensed the db2 wavelet coefficients of the 256×256 (raw) peppers image using $N = 17\,000$ measurements. Then we used various recovery algorithms to recover the wavelet coefficients, and we did the inverse wavelet transform to recover the original image.

For SuPrEM algorithms, we used a rate (3, 12) LDF with $M = 68\,000$ (the wavelet coefficients vector was padded with zeros to match the dimension). We set $L = 8000$ (the maximum sparsity the algorithm converged at) for SuPrEM. We ran the algorithm first with $\sigma = 0$. We also accommodated for noise, and estimated the per measurement noise to be $\sigma = 0.1\frac{\|\mathbf{r}\|_2}{\sqrt{N}}$ and ran the algorithm again. We ran our algorithms for just 50 iterations. For the reweighted SuPrEM algorithm, we let $\sigma = 0$ and we reweighed after 5 steps of the algorithm for a total of ten reweighings. For SMP, we used the SMP package [9]. We used a matrix with left degree 8 (which performed better than left degree 50 in this case) generated by this package, $L = 8000$, 50 iterations and convergence factor of 0.3. For the remaining methods, we used partial Fourier matrices whose rows were chosen randomly. For ℓ_1 with equality constraints, we used the L1MAGIC package. For LASSO, we used the GPSR package and $\tau = 0.001\|\mathbf{F}^T\mathbf{r}\|_\infty$, as described previously, and we thresholded the output to $L = 8000$ sparse coefficients and solved the appropriate least squares problem to get the final estimate. For CoSaMP and Subspace Pursuit, we used 100 iterations of the algorithm (and 150 iterations for the Richardson's iteration for calculating the least square solutions). For these algorithms, we used $L = 3000$ for CoSaMP, and $L = 3500$ for Subspace

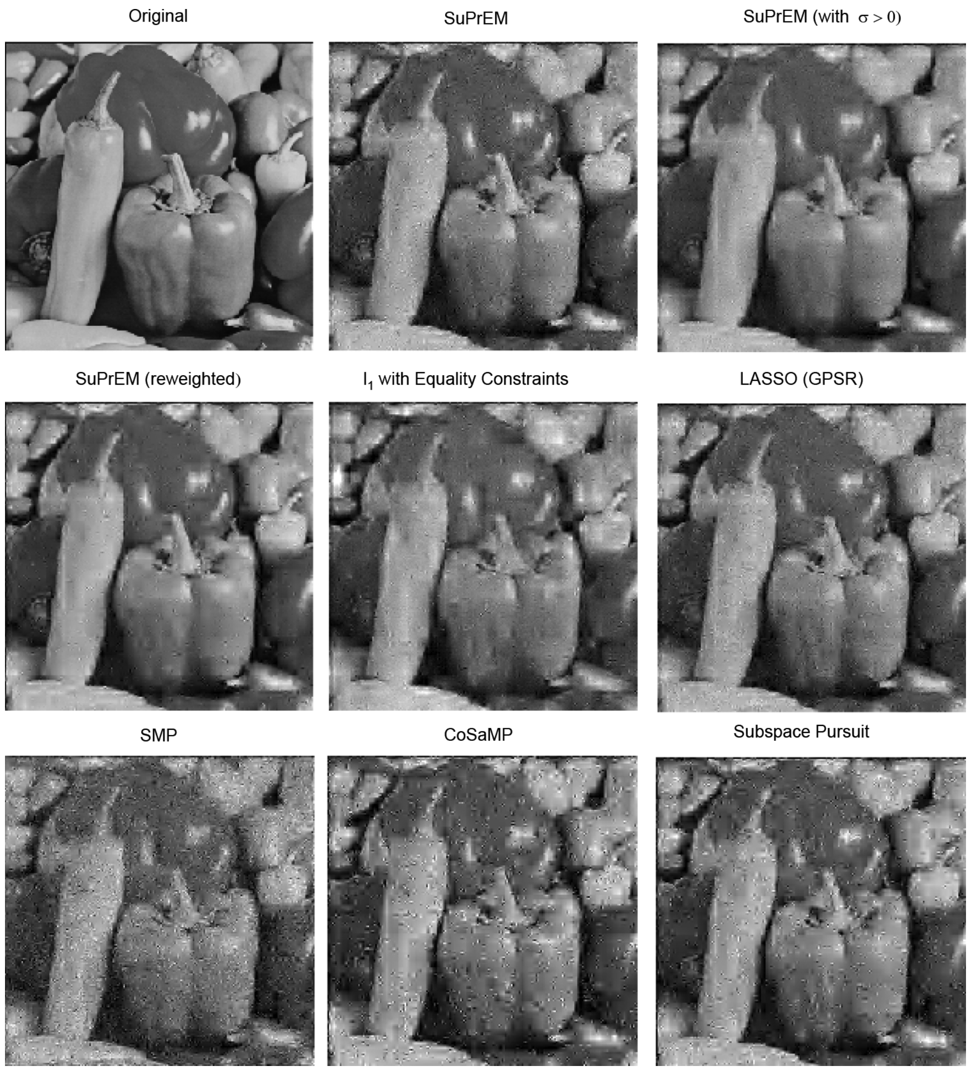


Fig. 4. Performance comparison of recovery algorithms with a 256×256 natural image whose db2 wavelet coefficients are compressively sensed with $N = 17\,000$ measurements.

Pursuit. These are slightly lower than the maximum sparsities they converged at ($L = 3500$ and $L = 4000$, respectively), but the values we used resulted in better visual quality and PSNR values. The results are depicted in Fig. 4.

The PSNR values for the methods are as follows: 23.41 dB for SuPrEM, 23.83 dB for SuPrEM (with nonzero σ^2), 24.79 for SuPrEM (reweighted), 20.18 dB for CoSaMP, 20.28 dB for SMP, 21.62 dB for ℓ_1, 23.61 dB for LASSO, 21.27 dB for subspace pursuit. Among the algorithms that assume no knowledge of noise, we see that SuPrEM outperforms the other algorithms both in terms of PSNR value and in terms of visual quality. The two algorithms that accommodate noise, SuPrEM and LASSO have similar PSNR values. Finally, the reweighted SuPrEM also assumes no knowledge of noise, and outperforms all other methods by about 1 dB and also in terms of visual quality, without requiring more running time.

E. Further Results

We studied the effect of the change of degree distributions. For a given $\frac{M}{N}$ ratio, we need to keep the ratio of $\frac{d_c}{d_v}$ fixed however the values can be varied. Thus, we compared the performance of $d_v = 3$ LDFs to $d_v = 5$ LDFs and observed that the latter actually performed sligthly better. However, having a higher d_v means more operations are required. We also observed that the number of iterations required for convergence was higher. Thus, we chose to use $d_v = 3$ LDFs that allowed faster decoding. We also note that increasing d_v too much (while keeping $\frac{M}{N}$ fixed) results in performance deterioration, since the graph becomes less sparse, and we run into shorter cycles which affect the performance of SPA.

We also tested the performance of our constructions and algorithms at $M = 100\,000$. With $\frac{L}{M}$ and $\frac{N}{M}$ fixed, interestingly the performance improves as $M \to \infty$ for Gaussian sparse signals for a fixed maximum number of 500 iterations. This is in line with intuitions drawn from Shannon theory [3]. Another interesting observation is that the number of iterations remain unchanged in this setting. In general, we observed that the number of iterations required for convergence is only a function of $\frac{L}{M}$ and does not change with M.

Finally, we also ran the simulations of Section IV-A for sparse signals whose nonzero coefficients have equal

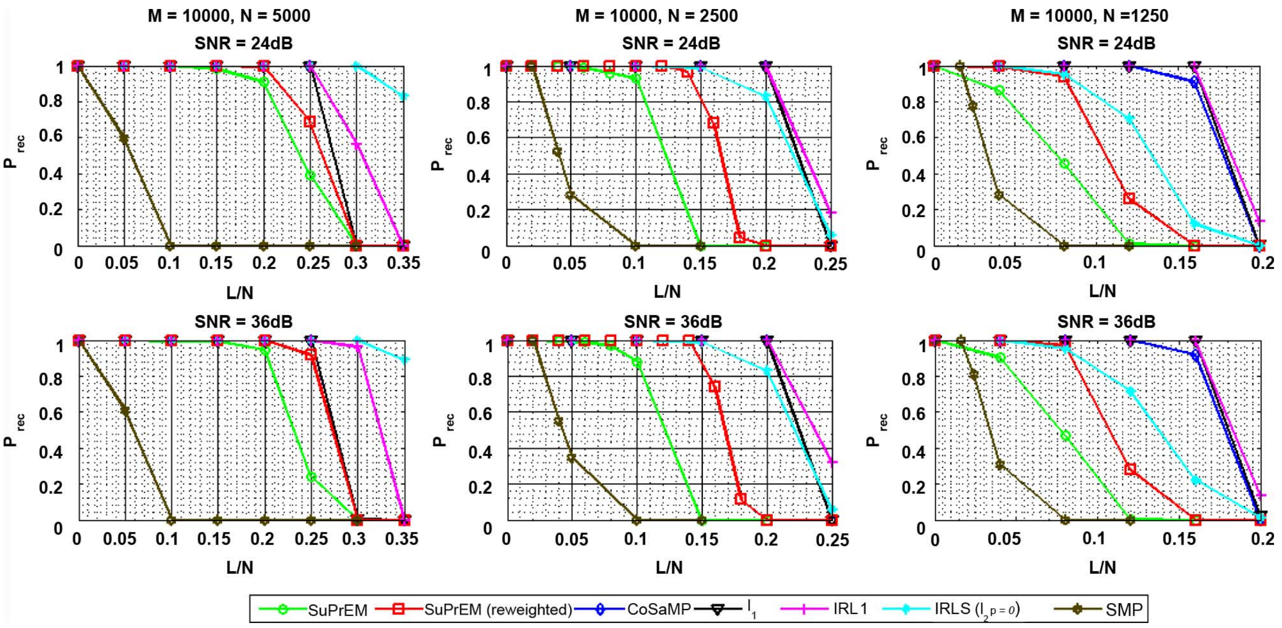


Fig. 5. Performance comparison of recovery algorithms for sparse signals whose nonzero coefficients have equal magnitude.

magnitude. These signals were generated to have nonzero coefficients with ± 1 values, and then normalized as described in Section IV-A for SNR values of 24 and 36 dB. Since even missing a single element in the support results in a large value of $\mathcal{D}_{\frac{e}{g}}(\mathbf{x}, \hat{\mathbf{x}}_e, \hat{\mathbf{x}}_{\text{genie}})$, we have used $P_{\text{rec}} = \frac{1}{K}\sum_{k=1}^{K} \mathbb{I}(\mathbf{x} \sim \hat{\mathbf{x}}_e)$, (where $\mathbf{x} \sim \hat{\mathbf{x}}_e$ is true only if $\text{supp}(\mathbf{x}) = \text{supp}(\hat{\mathbf{x}}_e)$) as our performance measure. The comparison algorithms were implemented as described in Section IV-B. The results are depicted in Fig. 5.

We note that there is an overall performance decrease for all methods at rates 2 and 4. In this setting, the SuPrEM algorithms are outperformed by CoSaMP and ℓ_1 regularization, which perform very close to each other. As with Gaussian signals, IRL1 outperforms ℓ_1 regularization as expected. The performance deterioration in SuPrEM is interesting for two reasons. First, sparse signals whose nonzero coefficients have equal magnitude correspond to finite alphabet signals, which is the standard setting in coding theory. If this information is present for the decoder, simply running SPA may result in a performance increase. Second, we note that the LDFs are binary as opposed to the Fourier transform matrices used for most of the other algorithms. This proves to be particularly challenging for the SuPrEM algorithms, since some check nodes will be equal (or very close) to zero, even though they are connected to nonzero components. This creates a problem on short cycles involving multiple nonzero variable nodes, since $d_v = 3$, and the messages on at least two incoming edges will be likely to declare these variable nodes to be close to zero. This may also be a reason why SuPrEM performs worse than IRLS in this setting. Further gains may be achieved by choosing the nonzero components of the LDFs over a continuous alphabet, for instance taking $\frac{N}{2}$ complex measurements, where the nonzero entries of the LDFs are chosen uniformly on the unit circle. We note this would also increase the storage requirements, and such modifications need further study.

V. Conclusion

In this paper, we constructed an ensemble of measurement matrices with small storage requirements. We denoted the members of this ensemble as LDFs. For these frames, we provided sparse reconstruction algorithms that have $O(Md_v^2)$ complexity and that are Bayesian in nature. We evaluated the performance of this ensemble of matrices and their decoding algorithms, and compared their performance to other state-of-the-art recovery algorithms and their associated measurement matrices. We observed that in various cases of interest, SuPrEM algorithms with LDFs provide excellent performance as noted in [25], outperforming other reconstruction algorithms with partial Fourier matrices. In particular, for Gaussian sparse signals and Gaussian noise, we are within 2-dB range of the theoretical lower bound in most cases.

There are various interesting research problems in this area. One is to find a deterministic message-passing schedule that performs as well as (or better than) our probabilistic message-passing schedule and that is amenable to analysis. The good empirical performance of LDFs, which are sparse matrices, and the associated SuPrEM algorithm may be used in sparse dictionary design algorithms, such as [38]. Adaptive measurements using the soft information available about the estimates, as well as online decoding (similar to Raptor codes [43]) is another open research area. Finally, if further information is available about the statistical properties of a class of signals (such as block-sparse signals or images represented on wavelet trees as in [6]), the decoding algorithms may be changed accordingly to improve performance.

A limitation of our study is that we are unable to analyze the performance of the iterative decoding algorithms for the LDFs theoretically. Such an analysis may in turn lead to useful design tools (like density evolution [39]) that might help with the construction of LDFs with irregular degree distributions. Along these lines, we also note the excellent works of [54], [55] that provide a density evolution-type analysis for noiseless compressive sensing of exactly sparse signals.

Appendix
Details on the Message-Passing Schedules

A message-passing schedule determines the order of messages passed between variable and check nodes of a factor graph. Traditionally, with LDPC codes, the so-called “flooding” schedule is used. In this schedule, at each iteration, all the variable nodes pass messages to their neighboring check nodes. Subsequently, all the check nodes pass messages to their neighboring variable nodes. For a cycle-free graph, SPA with a flooding schedule correctly computes *a posteriori* probabilities [13], [52]. An alternative schedule is the “serial” schedule, where we go through each variable node serially and compute the messages to the neighboring nodes. The order in which we go through variable nodes could be lexicographic, random or based on reliabilities.

In this section, we propose the following schedule based on the intuition derived from our simulations and results from LDPC codes [32], [52]: For the first iteration, all the check nodes send messages to variable nodes and vice-versa in a flooding schedule. After this iteration, with probability $\frac{1}{2}$ each check node is “on” or “off”. If a check node is off, it marks the edges connected to itself as an “inactive”, and sends back the messages it received to the variable nodes. If a check node is on, it marks the edges connected to itself as “active” and computes a new message. At the variable nodes, when calculating the new β, we only use the information coming from active edges. That is for $k = 1, 2, \dots, M$, let $\{k_1, k_2, \dots, k_{d_v}\}$ be the indices of the check nodes connected to the k^{th} variable node x_k. Let the incoming message from the check node r_{k_j} to the variable node x_k at the t^{th} iteration be $(\mu_{k_j}^{(t)}, \nu_{k_j}^{(t)})$ for $j = 1, \dots, d_v$. We will have

$$\lambda_k^{(t)} = \left(\sum_{(k,k_j) \text{ is an active edge}} \frac{1}{\nu_{k_j}^{(t)}} + \frac{1}{\beta_k^{(t-1)}} \right)^{-1},$$

$$\mu_k^{(t)} = \lambda_k^{(t)} \left(\sum_{(k,k_j) \text{ is an active edge}} \frac{\mu_{k_j}^{(t)}}{\nu_{k_j}^{(t)}} \right),$$

and

$$\beta_k^{(t)} = \frac{(\mu_k^{(t)})^2 + \lambda_k^{(t)}}{3}.$$

Thus, when there is no active edge, we do not perform a β update. For the special case when there is only one active edge (k, k_j), we let $\mu_k^{(t)} = \mu_{k_j}$. This is because the intrinsic information is more valuable, and the estimate on $\beta_k^{(t-1)}$ tends to be not as reliable. When we calculate the point estimate, we use all the information at the node, including the reliable and unreliable edges, i.e.,

$$\hat{V}_k^{(t)} = \left(\sum_{j=1}^{d_v} \frac{1}{\nu_{k_j}^{(t)}} + \frac{1}{\beta_k^{(t)}} \right)^{-1}$$

$$\hat{x}_k^{(t)} = \hat{V}_k^{(t)} \left(\sum_{j=1}^{d_v} \frac{\mu_{k_j}^{(t)}}{\nu_{k_j}^{(t)}} \right).$$

It is noteworthy that the flooding schedule and serial schedules tend to converge to local minima and they do not perform as well as this schedule we proposed.

References

[1] M. Akçakaya and V. Tarokh, “A frame construction and a universal distortion bound for sparse representations,” *IEEE Trans. Signal Process.*, vol. 56, no. 6, pp. 2443–2550, Jun. 2008.

[2] M. Akçakaya and V. Tarokh, “On sparsity, redundancy and quality of frame representations,” presented at the IEEE Int. Symp. Inf. Theory (ISIT), Nice, France, Jun. 2007.

[3] M. Akçakaya and V. Tarokh, “Shannon theoretic limits on noisy compressive sampling,” *IEEE Trans. Inf. Theory*, vol. 56, no. 1, pp. 492–504, Jan. 2010.

[4] D. Andrews and C. Mallows, “Scale mixtures of normal distributions,” *J. R. Stat. Soc.*, vol. 36, pp. 99–102, 1974.

[5] B. Babadi, N. Kalouptsidis, and V. Tarokh, “SPARLS: A low complexity recursive L1-regularized least squares algorithm,” *IEEE Trans. Signal Process.*, vol. 58, pp. 4013–4025, Aug. 2010.

[6] R. Baraniuk, V. Cevher, M. Duarte, and C. Hegde, “Model-based compressive sensing,” *IEEE Trans. Inf. Theory*, vol. 56, pp. 1982–2001, Apr. 2010.

[7] D. Baron, S. Sarvotham, and R. Baraniuk, “Bayesian compressive sensing via belief propagation,” *IEEE Trans. Signal Process.*, vol. 58, pp. 269–280, Jan. 2010.

[8] R. Berinde, A. C. Gilbert, P. Indyk, H. Karloff, and M. J. Strauss, “Combining geometry and combinatorics: A unified approach to sparse signal recovery,” presented at the Allerton Conf. Commun., Control, Comput., Monticello, IL, Sep. 2008.

[9] R. Berinde, P. Indyk, and M. Ruzic, “Practical near-optimal sparse recovery in the ell-1 norm,” presented at the Allerton Conf. Comm., Control, Comput., Monticello, IL, Sep. 2008.

[10] R. Berinde and P. Indyk, “Sequential sparse matching pursuit,” presented at the Allerton Conf. Commun., Control, Comput., Monticello, IL, Sep. 2009.

[11] D. Bickson, A. T. Ihler, and D. Dolev, “A low density lattice decoder via non-parametric belief propagation,” presented at the Allerton Conf. Commun., Control, Comput., Monticello, IL, Sep. 2009.

[12] J. M. Bioucas-Dias, “Bayesian wavelet-based image deconvolution: A GEM algorithm exploiting a class of heavy-tailed priors,” *IEEE Trans. Image Process.*, vol. 15, pp. 937–951, Apr. 2006.

[13] C. M. Bishop, *Pattern Recognition and Machine Learning*, 1st ed. New York: Springer, 2006.

[14] E. J. Candès, J. Romberg, and T. Tao, “Stable signal recovery for incomplete and inaccurate measurements,” *Commun. Pure Appl. Math.*, vol. 59, pp. 1207–1223, Aug. 2006.

[15] E. J. Candès and T. Tao, “Decoding by linear programming,” *IEEE Trans. Inf. Theory*, vol. 51, pp. 4203–4215, Dec. 2005.

[16] E. J. Candès, M. Wakin, and S. Boyd, “Enhancing sparsity by reweighted l1 minimization,” *J. Fourier Anal. Appl.*, vol. 14, pp. 877–905.

[17] R. Chartrand and W. Yin, “Iteratively reweighted algorithms for compressive sensing,” presented at the IEEE Int. Conf. Acoust., Speech, Signal Process. (IEEE ICASSP), Las Vegas, NV, Apr. 2008.

[18] T. Cormen, C. Leiserson, L. Rivest, and C. Stein, *Introduction to Algorithms*, 2nd ed. Cambridge, MA: MIT Press, 2001.

[19] W. Dai and O. Milenkovic, “Subspace pursuit for compressive sensing: Closing the gap between performance and complexity,” *IEEE Trans. Inf. Theory*, vol. 55, pp. 2230–2249, May 2009.

[20] D. L. Donoho, “Compressed sensing,” *IEEE Trans. Inf. Theory*, vol. 52, pp. 1289–1306, Apr. 2006.

[21] D. L. Donoho, A. Maleki, and A. Montanari, “Message passing algorithms for compressed sensing,” *Proc. Natl Acad. Sci.*, vol. 106, pp. 18914–18919, Nov. 2009.

[22] M. Figueiredo and R. Nowak, “An EM algorithm for wavelet-based image restoration,” *IEEE Trans. Image Process.*, vol. 12, pp. 906–916, Aug. 2003.

[23] M. A. T. Figueiredo and R. Nowak, “Wavelet-based image estimation: An empirical bayes approach using Jeffreys’ noninformative prior,” *IEEE Trans. Image Process.*, vol. 10, pp. 1322–1331, Sep. 2001.

[24] M. A. T. Figueiredo, R. D. Nowak, and S. J. Wright, “Gradient projection for sparse reconstruction: Application to compressed sensing and other inverse problems,” *IEEE J. Sel. Topics Signal Process.*, vol. 1, pp. 586–598, Dec. 2007.

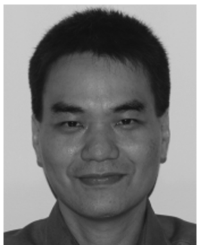
[25] A. Gilbert and P. Indyk, “Sparse recovery using sparse matrices,” *Proc. IEEE*, vol. 98, pp. 937–947, Jun. 2010.

[26] X.-Y. Hu, E. Eleftheriou, and D. M. Arnold, "Regular and irregular progressive edge-growth tanner graphs," *IEEE Trans. Inf. Theory*, vol. 51, pp. 386–398, Jan. 2005.
[27] P. Indyk and M. Ruzic, "Near-optimal sparse recovery in the ℓ_1 norm," presented at the IEEE Symp. Found. Comput. Sci., Philadelphia, PA, Oct. 2008.
[28] S. Ji, Y. Xue, and L. Carin, "Bayesian compressive sensing," *IEEE Trans. Signal Process.*, vol. 56, pp. 2346–2356, Jun. 2008.
[29] F. R. Kschischang, B. J. Frey, and H.-A. Loeliger, "Factor graphs and the sum-product algorithm," *IEEE Trans. Inf. Theory*, vol. 47, pp. 498–519, Feb. 2001.
[30] D. J. C. MacKay, "Good error correcting codes based on very sparse matrices," *IEEE Trans. Inf. Theory*, vol. 45, pp. 399–431, Mar. 1999.
[31] D. J. C. MacKay*, Information Theory, Inference and Learning Algorithms*, 3rd ed. Cambridge, U.K: Cambridge Univ. Press, 2002.
[32] Y. Mao and A. H. Banihashemi, "Decoding low-density parity-check codes with probabilistic scheduling," *IEEE Commun. Lett.*, vol. 5, pp. 414–416, Oct. 2001.
[33] A. Montanari, Graphical Models Concepts in Compressed Sensing [Online]. Available: http://arxiv.org/abs/1011.4328, preprint.
[34] T. K. Moon, "The EM algorithm in signal processing," *IEEE Signal Process. Mag.*, vol. 13, pp. 47–60, Nov. 1996.
[35] D. Needell and J. A. Tropp, "CoSaMP: Iterative signal recovery from incomplete and inaccurate samples," *Appl. Comp. Harmon. Anal.*, vol. 26, pp. 301–321, 2008.
[36] C. C. Paige and M. A. Saunders, "LSQR: Sparse linear equations and least squares problems," *ACM Trans. Math. Softw. (TOMS)*, vol. 8, pp. 195–209, Jun. 1982.
[37] J. Portilla, V. Strela, M. J. Wainwright, and E. P. Simoncelli, "Image denoising using scale mixtures of Gaussians in the wavelet domain," *IEEE Trans. Image Process.*, vol. 12, pp. 1338–1351, Nov. 2003.
[38] R. Rubinstein, M. Zibulevsky, and M. Elad, "Double sparsity: Learning sparse dictionaries for sparse signal approximation," *IEEE Trans. Signal Process.*, vol. 58, pp. 1553–1564, Mar. 2010.
[39] T. J. Richardson and R. L. Urbanke, "The capacity of low-density parity-check codes under message passing decoding," *IEEE Trans. Inf. Theory*, vol. 47, no. 2, pp. 599–618, Feb. 2001.
[40] C. Robert*, The Bayesian Choice: A Decision Theoretic Motivation*, 1st ed. New York, NY: Springer-Verlag, 1994.
[41] S. Sarvotham, D. Baron, and R. Baraniuk, "Sudocodes—fast measurement and reconstruction of sparse signals," presented at the IEEE Int. Symp. Inf. Theory (ISIT), Seattle, WA, Jul. 2006.
[42] S. Sarvotham, D. Baron, and R. Baraniuk, "Compressed sensing reconstruction via belief propagation," Electrical and Comput. Eng. Dept., .Rice Univ., Houston, TX, 2006.
[43] A. Shokrollahi, "Raptor codes," *IEEE Trans. Inf. Theory*, vol. 52, pp. 2551–2567, Jun. 2006.
[44] M. Sipser and D. A. Spielman, "Expander codes," *IEEE Trans. Inf. Theory*, vol. 42, pp. 1710–1722, Nov. 1996.
[45] R. M. Tanner, "A recursive approach to low complexity codes," *IEEE Trans. Inf. Theory*, vol. 27, pp. 533–547, Sep. 1981.
[46] M. E. Tipping, "Sparse bayesian learning and the relevance vector machine," *J. Mach. Learn. Res.*, vol. 1, pp. 211–244, 2001.
[47] M. E. Tipping, "Bayesian inference: An introduction to principles and practice in machine learning," in *Advanced Lectures on Machine Learning*. New York: Springer, 2004, pp. 41–62.
[48] E. van den Berg and M. P. Friedlander, "Probing the pareto frontier for basis pursuit solutions," *SIAM J. Sci. Comput.*, vol. 31, no. 2, pp. 890–912, 2008.
[49] M. J. Wainwright, "Sharp thresholds for noisy and high-dimensional recovery of sparsity using ℓ_1-constrained quadratic programming," *IEEE Trans. Inf. Theory*, vol. 55, pp. 2183–2202, May 2009.
[50] Y. Weiss and W. T. Freeman, "Correctness of belief propagation in Gaussian graphical models of arbitrary topology," *Neural Comput.*, vol. 13, no. 10, pp. 2173–2200, Oct. 2001.
[51] D. Wipf and S. Nagarajan, "Iterative reweighted ℓ_1 and ℓ_2 methods for finding sparse solutions," *IEEE J. Sel. Topics Signal Process.*, vol. 4, pp. 317–329, Apr. 2010.
[52] H. Xiao and A. H. Banihashemi, "Graph-based message-passing schedules for decoding LDPC codes," *IEEE Trans. Commun.*, vol. 52, pp. 2098–2105, Dec. 2004.
[53] W. Xu and B. Hassibi, "Efficient compressive sensing with deterministic guarantees using expander graphs," presented at the IEEE Inf. Theory Workshop, Lake Tahoe, CA, Sep. 2007.
[54] F. Zhang and H. D. Pfister, "On the iterative decoding of high rate LDPC codes with applications in compressed sensing," presented at the Allerton Conf. Commun., Control, Comput., Monticello, IL, Sep. 2008.
[55] F. Zhang and H. D. Pfister, "Verification decoding of high-rate LDPC codes with applications in compressed sensing," [Online]. Available: http://arxiv.org/abs/0903.2232, preprint.
[56] LDF Website [Online]. Available: http://people.fas.harvard.edu/~akcakaya/suprem.html

Mehmet Akçakaya received the Ph.D. degree from Harvard University, Boston, MA, in 2010.

He is currently a Research Fellow at the Beth Israel Deaconess Medical Center, Harvard Medical School. His research interests include the theory of sparse representations, compressed sensing and cardiovascular MRI.

Dr. Akçakaya was a finalist for the I. I. Rabi Young Investigator Award of the International Society of Magnetic Resonance in Medicine in 2011.



Jinsoo Park received the Bachelor's and Master's degrees in electronic/electrical engineering from Yonsei University, Seoul, Korea, in 1994 and 1996 and the Ph.D. degree in engineering sciences from Harvard University, Cambridge, MA, in 2009

From 1996 to 2002, he was an Engineer at Samsung, where he made contributions to the 3G wireless standards. His research interests include signal processing, graphical models, and network algorithms.

Vahid Tarokh received the Ph.D. degree from the University of Waterloo, Waterloo, ON, Canada, in 1995.

He is a Perkins Professor of applied mathematics and Hammond Vinton Hayes Senior Fellow of Electrical Engineering at Harvard University, Cambridge, MA. At Harvard, he teaches courses and supervises research in communications, networking and signal processing.

Dr. Tarokh has received a number of major awards and holds two honorary degrees.