Turbo Reconstruction of Structured Sparse Signals

Philip Schniter
Dept. ECE, The Ohio State University, Columbus OH 43210, schniter@ece.osu.edu

***Abstract*—This paper considers the reconstruction of structured-sparse signals from noisy linear observations. In particular, the support of the signal coefficients is parameterized by hidden binary pattern, and a structured probabilistic prior (e.g., Markov random chain/field/tree) is assumed on the pattern. Exact inference is discussed and an approximate inference scheme, based on loopy belief propagation (BP), is proposed. The proposed scheme iterates between exploitation of the observation-structure and exploitation of the pattern-structure, and is closely related to noncoherent turbo equalization, as used in digital communication receivers. An algorithm that exploits the observation structure is then detailed based on approximate message passing ideas. The application of EXIT charts is discussed, and empirical phase transition plots are calculated for Markov-chain structured sparsity.**[1]

I. Introduction

The problem of reconstructing sparse signals lies at the heart of many engineering and scientific applications. Here, the main objective is to estimate the sparse signal $\boldsymbol{x} \in \mathbb{C}^N$ from the noisy linear measurements $\boldsymbol{y} \in \mathbb{C}^M$,

$$\boldsymbol{y} \;=\; \boldsymbol{A}\boldsymbol{x} + \boldsymbol{w}, \tag{1}$$

where $\boldsymbol{A} \in \mathbb{C}^{M\times N}$ is a known matrix and $\boldsymbol{w} \in \mathbb{C}^M$ is additive noise, often modeled as circular white Gaussian, i.e., $\boldsymbol{w} \sim \mathcal{CN}(\boldsymbol{0}, \sigma^2\boldsymbol{I})$. By "sparse," we mean that the signal has only a few (say K, where $K \ll N$) non-zero coefficients.

In many cases of interest, the system of equations in (1) is underdetermined, i.e., $M \ll N$, so that, even in the noiseless case, there is no unique inverse. However, when $\boldsymbol{x}$ is known to be *sparse*, it is possible to accurately reconstruct $\boldsymbol{x}$ from $\boldsymbol{y}$ if the columns of $\boldsymbol{A}$ are sufficiently incoherent. For various sparse reconstruction algorithms, including convex-optimization-based, greedy, and iterative thresholding algorithms, there exist elegant bounds on reconstruction error that hold when $\boldsymbol{A}$ satisfies a certain *restricted isometry property* (RIP). (See [1] for a recent comprehensive overview.)

In many applications, however, the signal $\boldsymbol{x}$ has structure beyond simple sparsity. For example, the wavelet transform coefficients of natural scenes are not only approximately sparse, but also exhibit *persistence across scales* [1], which manifests as correlation within the sparsity pattern. Many other forms of structure in the sparsity pattern are also possible, and so we desire a powerful and flexible approach to modeling and exploiting such structure.

One approach to modeling sparsity structure is through the deterministic *union of subspaces* (UoS) approach. (See [1] for a recent comprehensive overview.) There, the K-sparse signal $\boldsymbol{x} \in \mathbb{C}^N$ is assumed to live in a UoS $\mathcal{M}_K = \cup_{m=1}^{m_K}\mathcal{X}_m$, where $\mathcal{X}_m$ is one of the $\binom{N}{K}$ canonical subspaces containing signals with support K. Two flavors of UoS that have garnered particular attention are *block-sparsity* and *tree-sparsity* [1]. For both, probabilistic guarantees of a model-based RIP have been derived and, from them, bounds on noisy reconstruction error [1]. The UoS approach has rather strong limitations, however. For example, it is not possible to specify whether it is *more* likely to find a signal component in one of the allowed subspaces $\mathcal{X}_m$ relative to another.

In this paper, we take a probabilistic approach to modeling sparsity structure, allowing the use of, e.g., *Markov chain* (MC), *Markov random field* (MRF), and *Markov tree* (MT) models [2]. Such models have been previously exploited for sparse reconstruction, but only to a limited extent. For example, [3] and [4] proposed Monte-Carlo-based [5] sparse reconstruction algorithms using MRF and MT models, respectively, and [6] and [7] proposed to iterate matching-pursuit with MAP pattern detection based on MRF and MT models, respectively. Monte-Carlo algorithms, while flexible, are typically regarded as computationally too expensive for many problems of interest. Matching-pursuit algorithms are typically much faster, but the schemes in [6], [7] are ad hoc.

We attack the problem of reconstructing structured-sparse signals through the framework of *belief propagation* (BP) [8]. While BP has been successfully used to recover *unstructured* sparse signals (e.g., [9], [10]), we believe that its application to structured sparse signals is novel. As we shall see, the BP framework suggests an *iterative* approach, where sparsity pattern beliefs are exchanged between two blocks, one exploiting observation structure and the other exploiting pattern structure. In this regard, our scheme resembles *turbo equalization* from digital communications [11], where bit beliefs are exchanged between a soft equalizer and a soft decoder. Our two blocks are themselves naturally implemented using BP, and we detail a particularly efficient algorithm based on the *approximate message passing* (AMP) framework recently proposed by Donoho, Maleki, and Montanari [10].

II. Structured Sparsity Model

Our structured-sparse signal model uses hidden binary indicators $\{s_n\}_{n=1}^N$, where $s_n \in \{0,1\}$. In particular, $s_n = 1$ indicates that the signal coefficient x_n is active (i.e., $x_n \neq 0$ w.p.1) while $s_n = 0$ indicates that x_n is inactive (i.e., $x_n = 0$ w.p.1). Assuming that the active signal coefficients are independently but non-identically distributed, we can write

$$p(x_n \,|\, s_n) \;=\; s_n q_n(x_n) + (1-s_n)\delta(x_n), \tag{2}$$

[1]This work was supported in part by the Office of Naval Research under grant N00014-07-1-0209.

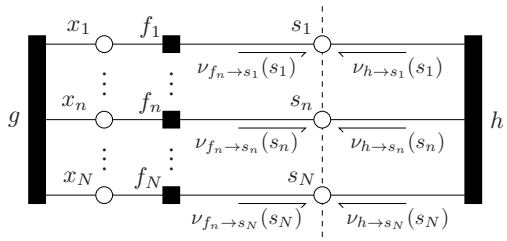

Fig. 1. Factor graph of posterior $p(\boldsymbol{x}, \boldsymbol{s} \,|\, \boldsymbol{y} = \boldsymbol{y}_0)$. The boxes represent constraint nodes, as defined in (4), and the circles represent variable nodes. The dashed line partitions the factor graph into two sub-graphs.

where $q_n(\cdot)$ denotes the pdf of x_n, when active, and $\delta(\cdot)$ denotes the Dirac delta. We refer to $\boldsymbol{s} = [s_1, s_2, \ldots, s_N]^T \in \{0,1\}^N$ as the *sparsity pattern*, and model structure in $\boldsymbol{s}$ through an assumed prior pmf $p(\boldsymbol{s})$. While we place no restrictions on $p(\boldsymbol{s})$, we note that Markov-chain/field/tree priors often lead to efficient inference algorithms [2].

III. Turbo Inference

Our primary goal is estimating the structured-sparse signal $\boldsymbol{x}$ given the observations $\boldsymbol{y} = \boldsymbol{y}_0$ in model (1). In particular, we are interested in computing minimum mean-squared error (MMSE) estimates of $\{x_n\}$.

A. Exact inference

Our estimation task is facilitated by the following factorization of the posterior pdf shown by the *factor graph* in Fig. 1.

$$p(\boldsymbol{x}, \boldsymbol{s} \,|\, \boldsymbol{y} = \boldsymbol{y}_0) \;\propto\; p(\boldsymbol{y} = \boldsymbol{y}_0 \,|\, \boldsymbol{x}, \boldsymbol{s}) p(\boldsymbol{x}, \boldsymbol{s}) \tag{3}$$

$$= \underbrace{p(\boldsymbol{s})}_{\triangleq\, h(\boldsymbol{s})} \underbrace{p(\boldsymbol{y} = \boldsymbol{y}_0 \,|\, \boldsymbol{x})}_{\triangleq\, g(\boldsymbol{x})} \prod_{n=1}^{N} \underbrace{p(x_n|s_n)}_{\triangleq\, f_n(x_n, s_n)}. \tag{4}$$

We use $\propto$ to denote equality after scaling to unit area.

The MMSE estimate of x_n is given by the mean of the marginal posterior $p(x_n \,|\, \boldsymbol{y} = \boldsymbol{y}_0)$, which can be written as

$$p(x_n \,|\, \boldsymbol{y} = \boldsymbol{y}_0) \;=\; \sum_{\boldsymbol{s}\in\{0,1\}^N} \int_{\boldsymbol{x}_{-n}} p(\boldsymbol{x}, \boldsymbol{s} \,|\, \boldsymbol{y} = \boldsymbol{y}_0) \tag{5}$$

$$\propto \sum_{s_n=0}^{1} f_n(x_n, s_n)\; p(s_n) \int_{\boldsymbol{x}_{-n}} g(\boldsymbol{x}) \prod_{q\neq n} \sum_{s_q=0}^{1} f_q(x_q, s_q) \times \sum_{\boldsymbol{s}_{-n,q}\in\{0,1\}^{N-2}} p(\boldsymbol{s}_{-n} \,|\, s_n), \tag{6}$$

where $\boldsymbol{z}_{-n}$ denotes vector $\boldsymbol{z}$ with the n^{th} element omitted, and $\boldsymbol{z}_{-n,q}$ denotes $\boldsymbol{z}$ with both the n^{th} and q^{th} elements omitted. Writing $p(\boldsymbol{s}_{-n} \,|\, s_n) = p(\boldsymbol{s}_{-n,q} \,|\, s_q, s_n) p(s_q \,|\, s_n)$, the last summation in (6) reduces to $p(s_q \,|\, s_n)$, giving

$$p(x_n \,|\, \boldsymbol{y} = \boldsymbol{y}_0) \;\propto\; \nu^{\text{exact}}_{f_n\to x_n}(x_n)\, \nu^{\text{exact}}_{g\to x_n}(x_n) \tag{7}$$

$$\nu^{\text{exact}}_{f_n\to x_n}(x_n) \;\triangleq\; \sum_{s_n=0}^{1} f_n(x_n, s_n) p(s_n) \tag{8}$$

$$\nu^{\text{exact}}_{g\to x_n}(x_n) \;\triangleq\; \int_{\boldsymbol{x}_{-n}} g(\boldsymbol{x}) \prod_{q\neq n} \sum_{s_q=0}^{1} f_q(x_q, s_q) p(s_q \,|\, s_n). \tag{9}$$

The notation $\nu^{\text{exact}}_{A\to B}(\cdot)$ will be explained in the sequel.

B. Approximate inference

Whereas exact posterior calculation via (7)-(9) is computationally prohibitive for typical problem sizes, approximate calculation can be efficiently accomplished using message passing [8], i.e., belief propagation (BP), on the factor graph in Fig. 1. In the sequel, we use $\nu_{A\to B}(\cdot)$ to denote a message passed from some node A to some adjacent node B in the factor graph. Our messages will either be pdfs on the real line or binary pmfs; which one will be clear from the context. BP proceeds according to the following two rules [8]: i) the message emitted by a variable node along a given edge equals the product of the messages coming into that variable node along the other edges, and ii) the message emitted by a function node along a given edge equals the integral of the product of the constraint function (associated with that node) and all messages coming into that function node along the other edges. (Here, "equals" holds after appropriate scaling.)

Using the framework of BP, the functions $\nu^{\text{exact}}_{f_n\to x_n}(\cdot)$ and $\nu^{\text{exact}}_{g\to x_n}(\cdot)$ from Section III-A can be approximated (up to a scaling factor) by steady-state versions of the messages

$$\nu^{(t)}_{f_n\to x_n}(x_n) \;\propto\; \sum_{s_n=0}^{1} f_n(x_n, s_n) \nu^{(t)}_{s_n\to f_n}(s_n) \tag{10}$$

$$\nu^{(t)}_{g\to x_n}(x_n) \;\propto\; \int_{\boldsymbol{x}_{-n}} g(\boldsymbol{x}) \prod_{q\neq n} \underbrace{\sum_{s_q=0}^{1} f_q(x_q, s_q) \nu^{(t)}_{s_q\to f_q}(s_q)}_{\nu^{(t)}_{f_q\to x_q}(x_n) \,=\, \nu^{(t)}_{x_q\to g}(x_n)}, \tag{11}$$

which depend on the other messages

$$\nu^{(t)}_{s_n\to f_n}(s_n) \;=\; \nu^{(t)}_{h\to s_n}(s_n) \tag{12}$$

$$\propto \sum_{\boldsymbol{s}_{-n}\in\{0,1\}^{N-1}} h(\boldsymbol{s}) \prod_{q\neq n} \underbrace{\nu^{(t-1)}_{s_q\to h}(s_q)}_{=\, \nu^{(t-1)}_{f_q\to s_q}(s_q)} \tag{13}$$

$$\nu^{(t)}_{f_n\to s_n}(s_n) \;\propto\; \int_{x_n} f_n(x_n, s_n) \underbrace{\nu^{(t)}_{x_n\to f_n}(x_n)}_{=\, \nu^{(t)}_{g\to f_n}(x_n)}. \tag{14}$$

We use the superscript-(t) to denote turbo iteration. These messages can then be combined for marginal inference:

$$\hat{p}^{(t)}(x_n \,|\, \boldsymbol{y} = \boldsymbol{y}_0) \;\propto\; \nu^{(t)}_{f_n\to x_n}(x_n)\, \nu^{(t)}_{g\to x_n}(x_n) \tag{15}$$

$$\hat{p}^{(t)}(s_n \,|\, \boldsymbol{y} = \boldsymbol{y}_0) \;\propto\; \nu^{(t)}_{f_n\to s_n}(s_n)\, \nu^{(t)}_{h\to s_n}(s_n), \tag{16}$$

where $\hat{p}^{(t)}$ denotes the iteration-t approximation to the pdf.

Because the factor graph in Fig. 1 contains many loops, exact inference is known to be NP-hard [12], and thus BP can only be claimed as an approximation. However, loopy BP has demonstrated very accurate results in similar settings (e.g., in LDPC and turbo decoding [13] and in inference on MRFs [14]). Furthermore, in the large-system limit (i.e., $M, N \to \infty$ with M/N fixed), BP has recently been shown to enjoy various optimality properties [10], [15].

With practical implementation in mind, we now partition our factor graph into the two sub-graphs separated by the dashed line in Fig. 1. The messages $\{\nu^{(t)}_{f_n\to s_n}(\cdot)\}_{n=1}^N$ form the outputs of the left sub-graph and the inputs to the right one, while the messages $\{\nu^{(t)}_{h\to s_n}(\cdot)\}_{n=1}^N$ form the outputs of the right sub-graph and the inputs to the left one. From this, we can interpret the BP scheme as iterating between two blocks, one which performs inference on the left sub-graph (which models structure in the observation) and the other which performs inference on the right sub-graph (which models structure in the sparsity pattern), with message-passing between blocks.

C. Relation to noncoherent turbo equalization

The iterative approach described in Section III-B mimics the *turbo equalization* [11] procedure that has become popular for the reception of digital communication signals, and in particular, *noncoherent turbo equalization* (e.g., [16]). There, the goal is to infer a sequence of information bits $\boldsymbol{b}$ that are transformed into coded bits $\boldsymbol{s}$, linearly modulated, and transmitted over a channel, yielding the noisy observations $\boldsymbol{y} = \boldsymbol{H}\boldsymbol{s} + \boldsymbol{w}$. In the noncoherent case, $\boldsymbol{H}$ depends on an *unknown* channel fading realization. Because optimal joint channel-estimation/equalization/decoding is computationally overwhelming, the bit inference task is often split into two subtasks, *soft noncoherent equalization* and *soft decoding*, which are then iterated. The equalizer calculates soft estimates of the coded bits $\boldsymbol{s}$ from $\boldsymbol{y}$ using knowledge of the channel structure and prior beliefs on $\boldsymbol{s}$ (as pilots or from the decoder). It then passes these $\boldsymbol{s}$-estimates to the decoder, which treats them as priors when exploiting its knowledge of the code structure to refine the $\boldsymbol{s}$-estimates. The decoder's estimates are then passed back to the equalizer, and so on, until they agree. To avoid self-reinforcement, the two blocks pass *extrinsic* information (i.e., a bit estimate may not employ the most recently assumed prior for that bit). When used in conjunction with powerful codes, turbo equalization facilitates practical communication at near-Shannon capacity [11].

To see the similarities between noncoherent turbo equalization and our BP-based approach to structured sparse signal recovery, we write the sparse coefficients as $x_n = \theta_n s_n$, where θ_n has prior pdf $q_n(\cdot)$, so that $\boldsymbol{x} = \mathcal{D}(\boldsymbol{\theta})\boldsymbol{s}$, where $\mathcal{D}(\boldsymbol{\theta})$ is the diagonal matrix constructed from $\boldsymbol{\theta} = [\theta_1, \theta_2, \ldots, \theta_N]^T$. In this case, the sparse observations (1) can be written as $\boldsymbol{y} = \boldsymbol{H}\boldsymbol{s} + \boldsymbol{w}$ with $\boldsymbol{H} = \boldsymbol{A}\,\mathcal{D}(\boldsymbol{\theta})$. Thus, the structured sparsity pattern $\boldsymbol{s}$ is analogous to "coded bits," the active-coefficient vector $\boldsymbol{\theta}$ is analogous to the "unknown fading realization," and $\boldsymbol{A}$ contributes known structure to the unknown "channel matrix" $\boldsymbol{H}$. Inference on the left sub-graph of Fig. 1 is analogous to "soft noncoherent equalization," while that on the right sub-graph corresponds to "soft decoding."

Taking cues from turbo equalization, we will henceforth refer to inference on the left sub-graph of Fig. 1 as "*sparsity pattern equalization*" (SPE) and inference on the right sub-graph as "*sparsity pattern decoding*" (SPD). We now formally decouple these subtasks and represent each of them using a separate factor graph, as in Fig. 2. For this, we define two

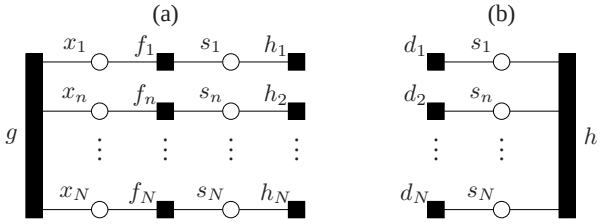

Fig. 2. Decoupling of partitioned factor graph from Fig. 1 into (a) sparsity pattern equalization and (b) sparsity pattern decoding.

additional t^{th} turbo-iteration constraint functions,

$$h^{(t)}_n(s_n) \triangleq \nu^{(t)}_{h\to s_n}(s_n) \tag{17}$$

$$d^{(t)}_n(s_n) \triangleq \nu^{(t-1)}_{f_n\to s_n}(s_n), \tag{18}$$

recalling (13)-(14), initialized using $\nu^{(0)}_{f_n\to s_n}(s_n) = 0.5$.

For t^{th}-iteration SPE, (16) implies an output message $\nu^{(t)}_{f_n\to s_n}(s_n) \propto \hat{p}^{(t)}(s_n|\boldsymbol{y} = \boldsymbol{y}_0)/h^{(t)}_n(s_n)$, which equals the (approximate) likelihood of s_n, since $h^{(t)}_n(\cdot)$ acts as SPE's prior on s_n. As such, the SPE output messages are extrinsic. Similar reasoning applies to SPD outputs as well.

IV. Sparsity Pattern Equalization

As previously discussed, SPE accepts independent but non-identical probabilities on the indicators $\{s_n\}$ and returns likelihoods on the same. One can imagine a number of ways to accomplish this task, e.g., Monte Carlo methods [5], "soft" Bayesian matching pursuit [17], expectation maximization, or BP. Below we outline a BP-based technique that follows the "approximate message passing" (AMP) framework recently proposed by Donoho, Maleki, and Montanari [10], [18]. Since we focus on a single turbo iteration t, we suppress the superscript-(t) notation on messages in this section.

For BP-based SPE, we expand the g node in Fig. 2(a), yielding the loopy factor graph in Fig. 3, with constraints

$$g_m(\boldsymbol{x}) \triangleq \mathcal{CN}(y_m; \boldsymbol{a}_m^H\boldsymbol{x}, \sigma^2), \tag{19}$$

where $\boldsymbol{a}_m^H$ denotes the m^{th} row of $\boldsymbol{A}$. Noting that SPE will require several iterations of message passing between nodes $\{g_m\}$ and $\{x_n\}$, we will henceforth use $\nu^i_{x_n\to g_m}$ and $\nu^i_{g_m\to x_n}$ to denote the SPE-iteration-i messages. In addition, we will assume Gaussian active-coefficients, i.e.,

$$q_n(x_n) = \mathcal{CN}(x_n; 0, \sigma_n^2), \tag{20}$$

though other distributions could be handled using similar techniques. Since (20) allows non-uniform coefficient variance, we can assume w.l.o.g that all columns of $\boldsymbol{A}$ have unit ℓ_2-norm (after absorbing any variations into $\{\sigma_n^2\}$). In the sequel, we use λ_n to abbreviate $h_n(1)$, the prior probability of $s_n = 1$ assumed by SPE. Thus, the coefficient prior implicit to our incarnation of AMP is Bernoulli-Gaussian, with the form

$$\nu_{f_n\to x_n}(x_n) = \lambda_n\mathcal{CN}(x_n; 0, \sigma_n^2) + (1-\lambda_n)\delta(x_n). \tag{21}$$

In the sequel, we will make use of the abbreviations

$$\alpha_n(c) \triangleq \tfrac{\sigma_n^2}{c+\sigma_n^2}, \quad \beta_n(c) \triangleq \tfrac{1-\lambda_n}{\lambda_n}\tfrac{c+\sigma_n^2}{c}, \quad \zeta_n(c) \triangleq \tfrac{\sigma_n^2}{c(c+\sigma_n^2)}.$$

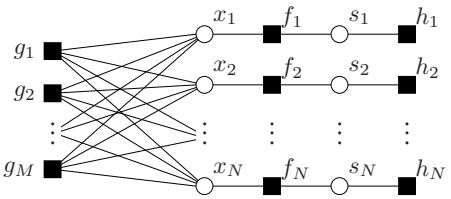

Fig. 3. Factor graph for BP-based implementation of SPE.

A. BP approximation via the large-system limit

Due to the mixture prior (21), exact calculation of $\nu^i_{g_m\to x_n}(x_n)$ would involve the integration of 2^{N-1} terms, which is clearly impractical. However, in the *large system limit* (i.e., $M, N \to \infty$ with M/N fixed), the central limit theorem motivates the treatment of $\nu^i_{g_m\to x_n}(x_n)$ as Gaussian [18]. In this case, it is sufficient to parameterize the inputs to g_m via

$$\mu^i_{nm} \triangleq \int_{x_n} x_n \nu^i_{x_n\to g_m}(x_n), \quad v^i_{nm} \triangleq \int_{x_n} (x_n - \mu^i_{nm})^2 \nu^i_{x_n\to g_m}(x_n), \tag{22}$$

which yields outputs from g_m that take the form

$$\nu^i_{g_m\to x_n}(x_n) \propto \mathcal{CN}(A_{mn}x_n; z^i_{mn}, c^i_{mn}) \tag{23}$$
$$z^i_{mn} \triangleq y_m - \textstyle\sum_{q\neq n} A_{mq}\mu^i_{qm} \tag{24}$$
$$c^i_{mn} \triangleq \sigma^2 + \textstyle\sum_{q\neq n} |A_{mq}|^2 v^i_{qm} \tag{25}$$

as can be shown using the fact that

$$\prod_q \mathcal{CN}(x;\mu_q,v_q) \propto \mathcal{CN}\left(x; \frac{\sum_q \mu_q/v_q}{\sum_q v_q^{-1}}, \frac{1}{\sum_q v_q^{-1}}\right). \tag{26}$$

From (22), we see that μ^{i+1}_{nm} and v^{i+1}_{nm} are then determined by the mean and variance, respectively, of the pdf

$$\nu^{i+1}_{x_n\to g_m}(x_n) \propto \nu_{f_n\to x_n}(x_n) \textstyle\prod_{l\neq m} \nu^i_{g_l\to x_n}(x_n). \tag{27}$$

Using (26), the product term in (27) reduces to

$$\mathcal{CN}\left(x_n; \frac{\sum_{l\neq m} A^*_{ln} z^i_{ln}/c^i_{ln}}{\sum_{l\neq m} |A_{lm}|^2/c^i_{ln}}, \frac{1}{\sum_{l\neq m} |A_{ln}|^2/c^i_{ln}}\right), \tag{28}$$

and so, under the large-system-limit approximations

$$c^i_{ln} \approx c^i_n \triangleq \tfrac{1}{M}\textstyle\sum_{m=1}^M c^i_{mn} \tag{29}$$

and $\sum_{l\neq m} |A_{ln}|^2 \approx \sum_{l=1}^M |A_{ln}|^2 = 1$, (27) simplifies to

$$\nu^{i+1}_{x_n\to g_m}(x_n) \propto \big(\lambda_n \mathcal{CN}(x_n;0,\sigma^2_n) + (1-\lambda_n)\delta(x_n)\big) \times \mathcal{CN}\big(x_n; \textstyle\sum_{l\neq m} A^*_{ln} z^i_{ln}, c^i_n\big). \tag{30}$$

Applying (26) to (30), we find, after some algebra, that

$$\mu^{i+1}_{nm} = \alpha_n(c^i_n)\theta^i_{nm}/(1+\gamma^i_{nm}) \tag{31}$$
$$v^{i+1}_{nm} = \gamma^i_{nm}|\mu^{i+1}_{nm}|^2 + \mu^{i+1}_{nm} c^i_n/\theta^i_{nm} \tag{32}$$
$$\theta^i_{nm} \triangleq \textstyle\sum_{l\neq m} A^*_{ln} z^i_{ln} \tag{33}$$
$$\gamma^i_{nm} \triangleq \beta_n(c^i_n) \exp\big(-\zeta_n(c^i_n)\,|\theta^i_{nm}|^2\big). \tag{34}$$

For the first turbo iteration, we desire that $\nu^0_{g_m\to x_n}(x_n) \propto 1$, and so we set $c^0_n \gg \sigma^2_n$ and $z^0_{mn} = y_m$ for all n. For second and later turbo iterations, we set $\{c^0_n\}$ and $\{z^0_{mn}\}$ equal to their final values from the previous turbo iteration.

The i^{th} SPE iteration yields the x_n-posterior approximation

$$\hat{p}^i(x_n \,|\, \boldsymbol{y} = \boldsymbol{y}_0) \propto \nu_{f_n\to x_n}(x_n) \textstyle\prod_{l=1}^M \nu^{i-1}_{g_l\to x_n}(x_n). \tag{35}$$

The mean and variance of (35) constitute the MMSE estimate of x_n and its MSE. Noting that (35) differs from (27) only in the inclusion of the m^{th} product term, analysis similar to (28)-(34) yields the following iteration-$(i+1)$ MMSE quantities:

$$\mu^{i+1}_n \triangleq \hat{\mathrm{E}}^i\{x_n \,|\, \boldsymbol{y} = \boldsymbol{y}_0\} = \alpha_n(c^i_n)\theta^i_n/(1+\gamma^i_n) \tag{36}$$
$$v^{i+1}_n \triangleq \widehat{\mathrm{var}}^i\{x_n \,|\, \boldsymbol{y} = \boldsymbol{y}_0\} = \gamma^i_n|\mu^{i+1}_n|^2 + \mu^{i+1}_n c^i_n/\theta^i_n \tag{37}$$
$$\theta^i_n \triangleq \textstyle\sum_{l=1}^M A^*_{ln} z^i_{ln} \tag{38}$$
$$\gamma^i_n \triangleq \beta_n(c^i_n) \exp\big(-\zeta_n(c^i_n)\,|\theta^i_n|^2\big). \tag{39}$$

The i^{th} SPE iteration approximation to the s_n-posterior is

$$\hat{p}^i(s_n \,|\, \boldsymbol{y} = \boldsymbol{y}_0) \propto \nu^{i-1}_{f_n\to s_n}(s_n)\, \nu_{h_n\to s_n}(s_n), \tag{40}$$

where $\nu^i_{f_n\to s_n}(s_n) \propto \displaystyle\int_{x_n} f_n(x_n,s_n) \prod_{l=1}^M \nu^i_{g_l\to x_n}(x_n)$. (41)

From $f_n(x_n,s_n) = s_n\mathcal{CN}(x_n;0,\sigma^2_n) + (1-s_n)\delta(x_n)$, it is easy to show that the output log-likelihood ratio (LLR) is

$$L^i_n \triangleq \ln \frac{\nu^i_{f_n\to s_n}(1)}{\nu^i_{f_n\to s_n}(0)} = \ln \frac{c^i_n}{c^i_n+\sigma^2_n} + \zeta_n(c^i_n)\,|\theta^i_n|^2. \tag{42}$$

B. Approximate message passing

The approximate BP algorithm outlined in Section IV-A updates $\mathcal{O}(NM)$ variables per iteration: $\{z^i_{mn}\}$, $\{\mu^i_{nm}\}$, $\{v^i_{nm}\}$, and $\{c^i_n\}$ for $m = 1\ldots M$ and $n = 1\ldots N$. When N and M are large, the resulting complexity may be undesirably high, motivating us to find a simpler scheme.

Recently, Donoho, Maleki and Montanari proposed a family of so-called *first-order approximate message passing* (AMP) algorithms [10], [18] that greatly simplify BP algorithms of the form outlined in Section IV-A by tracking only $\mathcal{O}(N)$ variables. In particular, [18] describes how the AMP approach can be applied to generic independent (but not necessarily identical) real-coefficient priors. Paraphrasing [18, Sec. V], AMP initializes $\mu^0_n = 0$, $z^0_m = y_m$, and $c^0 \gg \sigma^2_n$ for all n and m, and then iterates (43)-(47) for $i = 0, 1, 2, \ldots$

$$\theta^i_n = \textstyle\sum_{m=1}^M A^*_{mn} z^i_m + \mu^i_n \tag{43}$$
$$\mu^{i+1}_n = F_n(\theta^i_n; c^i) \tag{44}$$
$$v^{i+1}_n = G_n(\theta^i_n; c^i) \tag{45}$$
$$c^{i+1} = \sigma^2 + \tfrac{1}{M}\textstyle\sum_{n=1}^N v^{i+1}_n \tag{46}$$
$$z^{i+1}_m = y_m - \textstyle\sum_{n=1}^N A_{mn}\mu^{i+1}_n + \frac{z^i_m}{M}\sum_{n=1}^N F'_n(\theta^i_n; c^i). \tag{47}$$

Above, $F_n(.;.)$, $G_n(.;.)$, and $F'_n(.;.)$ are nonlinear functions that depend on the coefficient prior. (See [18] for definitions.)

Using analyses similar to those in Section IV-A, one can show that, for the Bernoulli-Gaussian prior (21), these nonlin-

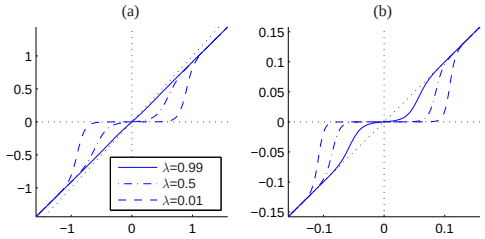

Fig. 4. The Bernoulli-Gaussian soft thresholding function $F_n(\theta;c)$ versus θ for several values of λ_n and for (a) $c=0.1$ and (b) $c=0.001$.

ear functions take the following form:

$$F_n(\theta;c) \;=\; \frac{\alpha_n(c)}{1+\beta_n(c)e^{-\zeta_n(c)|\theta|^2}}\theta \tag{48}$$

$$G_n(\theta;c) \;=\; \beta_n(c)e^{-\zeta_n(c)|\theta|^2}\,|F_n(\theta;c)|^2 + \frac{c}{\theta}F_n(\theta;c) \tag{49}$$

$$F'_n(\theta;c) \;=\; \frac{\alpha_n(c)}{1+\beta_n(c)e^{-\zeta_n(c)|\theta|^2}} \times\left[1+\frac{\zeta_n(c)|\theta|^2}{1+(\beta_n(c)e^{-\zeta_n(c)|\theta|^2})^{-1}}\right]. \tag{50}$$

The function $F_n(.;.)$ plays the role of a soft threshold in (44), suppressing small values in the projected residual θ_n^i, and passing large ones. (See Fig. 4.) Here "small" and "large" are determined by the values of λ_n and c^i, where the latter has be observed to decrease with SPE iteration i.

AMP (43)-(50) follows from the approximate BP algorithm of Section IV-A via the intuition that, in the large system limit, both the n dependence of z_{mn}^i in (24) and the m dependence of θ_{nm}^i in (33) are weak. However, it is not sufficient to simply ignore these dependencies. Comparing (43) to (33), we see that the extra term $A_{mn}^* z_m^i$ is offset by μ_n^i, and comparing (47) to (24), we see that the extra term $A_{mn}\mu_n^{i+1}$ is offset by the last term in (47), which is analogous to the Onsager reaction term from statistical physics [10]. As is evident from above, the complexity of AMP is dominated by the two matrix multiplies (43) and (47) which, in some applications, can be efficiently implemented using a fast algorithm like the FFT.

V. Sparsity Pattern Decoding

The implementation of SPD depends strongly on the structure of $p(\boldsymbol{s})$. When $p(\boldsymbol{s})$ has a Markov structure, the inference task is a well-studied one that can be efficiently implemented using message passing algorithms [8], [13], [14], in some cases (e.g., with Markov chains and trees) optimally [2].

VI. EXIT charts

As discussed earlier, BP-based implementations of SPE and of SPD are expected to work very well on their own. The question remains: How well do SPE and SPD work as a pair? In the turbo literature, *extrinsic information transfer (EXIT) charts* [19] are used to predict the extent to which two soft-input/output blocks can help each other in decoding. The same ideas can be used to predict the extent to which SPE and SPD can help each other learn the sparsity pattern.

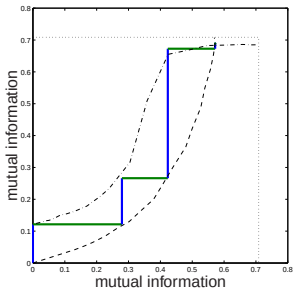

Fig. 5. EXIT chart, for $N=512$, $M=128$, $\lambda=0.2$, $\gamma=0.2$, and SNR$=30$ dB, showing SPE EXIT curve (top), SPD EXIT curve (bottom), turbo trajectory (middle), and i.i.d entropy rate (dotted).

The *SPE EXIT curve* plots the mutual information between output LLRs $\{L_n^i\}$ and true pattern $\{s_n\}$, versus the mutual information between soft inputs $\{\ln\frac{\lambda_n}{1-\lambda_n}\}$ and true pattern $\{s_n\}$. Similarly, the *SPD EXIT curve* plots SPD's output mutual information versus its input mutual information. From these two input-output maps, one can predict whether the information about $\{s_n\}$ can be increased through another turbo iteration, or whether learning has saturated. Sparsity-pattern EXIT charts differ from conventional EXIT charts in one small detail: the mutual information is plotted on the interval $[0, I_{\max}]$, where $I_{\max}$ is generally less than 1. This is because the entropy rate of i.i.d $\{s_n\}$ is <1 when $\{s_n\}$ is non-equiprobable, as occurs in the sparse problem setting.

Figure 5 shows an example EXIT chart. (The simulation setup will be described in Section VII.) The turbo trajectory, starting in the lower left corner (with zero information), moves up with every application of SPE, and to the right with every application of SPD, progressing towards the upper right corner (representing perfect pattern recovery). The trajectory is (approximately) bounded above by the SPE EXIT curve, and (approximately) bounded to the right by the SPD EXIT curve, and thus cannot progress beyond the point at which those curves intersect. In this example, the SPE/SPD together do not allow perfect pattern recovery, though they come close.

VII. Numerical experiments

Numerical experiments were conducted for the observation model (1), where the elements of $\boldsymbol{A}$ were independently drawn from a $\mathcal{CN}(0,M^{-1})$ distribution and where the signal coefficients were generated via $p(x_n\,|\,s_n)=s_n\mathcal{CN}(x_n;0,1)+(1-s_n)\delta(x_n)$ using a Markov chain (MC)-generated binary sparsity pattern $\{s_n\}$. Such a MC is fully described by the transition probabilities $p_{01}\triangleq\Pr\{s_n=0\,|\,s_{n-1}=1\}$ and $p_{10}\triangleq\Pr\{s_n=1\,|\,s_{n-1}=0\}$, yielding a stationary distribution with *activity rate* $\lambda\triangleq\Pr\{s_n=1\}=(1+p_{01}/p_{10})^{-1}$. To generate patterns with a desired $\lambda\in(0,1]$, we set $p_{01}=p_{10}(\lambda^{-1}-1)$ with $p_{10}=\gamma\lambda$, for $\gamma\in(0,1]$ called the *Markov independence parameter*. Note that, as γ increases, the pattern becomes less correlated, with $\gamma=1$ corresponding to an i.i.d pattern. In fact, the pattern's coherence time T is inversely proportional to γ, i.e., $T=(2\gamma\lambda(1-\lambda))^{-1}$. To ensure that we used only "typical" pattern realizations, we

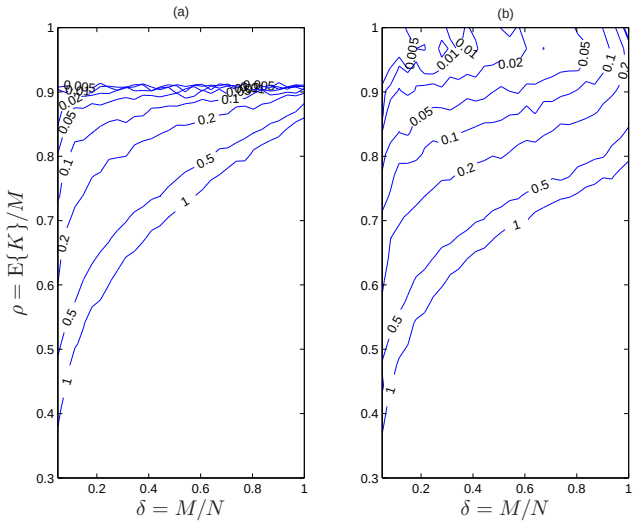

Fig. 6. Empirical 50%phase transition curves for the events (a) NMSE ≤ -20dB and (b) NMSE $\leq$ NMSEgenie $+$ 1dB, for various values of the Markov independence parameter γ (shown by the labels on each curve).

discarded any $\{s_n\}$ for which $\hat{\lambda} \triangleq \frac{1}{N}\sum_{n=1}^{N} s_n$ differed from λ by more than 5%. All experiments used $N = 512$ and σ^2 such that SNR $= 30$dB, where SNR $\triangleq \mathrm{E}\{\|\boldsymbol{A}\boldsymbol{x}\|_2^2\}/\mathrm{E}\{\|\boldsymbol{w}\|_2^2\}$.

SPD was implemented using the standard forward-backward (or sum-product) algorithm [8], [13], which performs exact inference for our MC pattern, and SPE was implemented using the AMP technique outlined in Section IV-B, allowing 10 SPE iterations. The proposed turbo scheme was allowed 5 iterations. We measured performance using NMSE $\triangleq \mathrm{E}\{\|\hat{\boldsymbol{x}} - \boldsymbol{x}_0\|_2^2\}/\mathrm{E}\{\|\boldsymbol{x}_0\|_2^2\}$, where $\boldsymbol{x}_0$ denotes the true parameter vector and $\hat{\boldsymbol{x}}$ denotes the final turbo estimate. Average performance was tested under different problem settings, parameterized by γ, the *undersampling ratio* $\delta \triangleq M/N$, and the *normalized sparsity* $\rho \triangleq \mathrm{E}\{K\}/M = \lambda N/M$, where K denotes the number of non-zero coefficients in $\{x_k\}$ (and $\{s_n\}$).

In Fig. 6(a), for various choices of Markov independence parameter γ, we show *empirically estimated phase transition* curves [10] for the event NMSE ≤ -20dB, labeling each curve with the corresponding value of γ. The curves were constructed as follows. Defining NMSE ≤ -20dB as "success," we empirically estimated the probability of success at each triplet (γ, δ, ρ) by comparing the NMSE for 200 independent realizations of $(\boldsymbol{A}, \boldsymbol{x}, \boldsymbol{s}, \boldsymbol{w})$. Matlab's `contour` command was then used to draw the 50% success-probability contour, which always manifested as a curve bisecting the (δ, ρ) space: points northwest of the curve had $< 50\%$ success, and points southeast of the curve had $> 50\%$ success. Figure 6(a) shows that, as γ decreases (i.e., the sparsity becomes more structured), the successful-(δ, ρ) domain expands. This domain is bounded above by $\rho \approx 0.91$, however, since above that line the NMSE calculated under *genie-aided* perfect sparsity-pattern knowledge itself exceeded -20dB. Thus, in Fig. 6(b), we show 50% phase transition curves for the event NMSE $\leq$ NMSEgenie$+1$dB. There, the successful-(δ, ρ) domain appears to expand with every decrease in γ, i.e., increase in structure.

VIII. Conclusion

In this paper, we considered the reconstruction of structured-sparse signals from noisy linear observations, using probabilistic priors to model sparsity structure. Exact inference was examined and found to be computationally impractical, and so an approximate inference scheme, based on belief propagation, was proposed that iterates between exploitation of observation structure (via "SPE") and sparsity pattern structure (via "SPD"), with connections to noncoherent turbo equalization. Using the AMP framework recently proposed by Donoho, Maleki, and Montanari, an efficient implementation of SPE was outlined, and EXIT charts were proposed as a means of predicting the interaction between SPE and SPD. Finally, performance was quantified using NMSE-based empirical phase transition curves, demonstrating that sparsity pattern structure can be successfully exploited by our turbo scheme.

References

[1] R. G. Baraniuk, V. Cevher, M. F. Duarte, and C. Hegde, "Model-based compressive sensing," *IEEE Trans. Inform. Theory*, vol. 56, pp. 1982–2001, Apr. 2010.

[2] C. A. Bouman, "Markov random fields and stochastic image models," in *IEEE Int. Conf. Image Processing Tutorial*, Oct. 1995.

[3] P. J. Wolfe, S. J. Godsill, and W.-J. Ng, "Bayesian variable selection and regularization for time-frequency surface estimation," *J. R. Statist. Soc. B*, vol. 66, no. 3, pp. 575–589, 2004.

[4] L. He and L. Carin, "Exploiting structure in wavelet-based Bayesian compressive sensing," *IEEE Trans. Signal Process.*, vol. 57, pp. 3488–3497, Sept. 2009.

[5] C. P. Robert and G. Casella, *Monte Carlo Statistical Methods*. New York: Springer, 2nd ed., 2004.

[6] V. Cevher, M. F. Duarte, C. Hedge, and R. G. Baraniuk, "Sparse signal recovery using Markov random fields," in *Proc. Neural Inform. Process. Syst. Conf.*, (Vancouver, B.C.), Dec. 2008.

[7] M. F. Duarte, M. B. Wakin, and R. G. Baraniuk, "Wavelet-domain compressive signal recontruction using a hidden Markov tree model," in *Proc. IEEE Int. Conf. Acoust. Speech & Signal Process.*, (Las Vegas, NV), pp. 5137–5140, Apr. 2008.

[8] J. Pearl, *Probabilistic Reasoning in Intelligent Systems*. San Mateo, CA: Morgan Kaufman, 1988.

[9] D. Baron, S. Sarvotham, and R. G. Baraniuk, "Bayesian compressive sensing via belief propagation," *IEEE Trans. Signal Process.*, vol. 58, pp. 2269–280, Jan. 2010.

[10] D. L. Donoho, A. Maleki, and A. Montanari, "Message passing algorithms for compressed sensing," *Proc. National Academy of Sciences*, vol. 106, pp. 18914–18919, Nov. 2009.

[11] R. Koetter, A. C. Singer, and M. Tüchler, "Turbo equalization," *IEEE Signal Process. Mag.*, vol. 21, pp. 67–80, Jan. 2004.

[12] G. F. Cooper, "The computational complexity of probabilistic inference using Bayesian belief networks," *Artificial Intelligence*, vol. 42, 1990.

[13] D. J. C. MacKay, *Information Theory, Inference, and Learning Algorithms*. New York: Cambridge University Press, 2003.

[14] W. T. Freeman, E. C. Pasztor, and O. T. Carmichael, "Learning low-level vision," *Intl. J. Computer Vision*, vol. 40, pp. 25–47, Oct. 2000.

[15] D. Guo, D. Baron, and S. Shamai, "A single-letter characterization of optimal noisy compressed sensing," in *Proc. Allerton Conf. Commun. Control Comput.*, (Monticello, IL), Oct. 2009.

[16] S.-J. Hwang and P. Schniter, "Efficient multicarrier communication for highly spread underwater acoustic channels," *IEEE J. Sel. Areas Commun.*, vol. 26, pp. 1674–1683, Dec. 2008.

[17] P. Schniter, L. C. Potter, and J. Ziniel, "Fast Bayesian matching pursuit," in *Proc. Inform. Theory & Appl. Workshop*, Feb. 2008.

[18] D. L. Donoho, A. Maleki, and A. Montanari, "Message passing algorithms for compressed sensing: I. Motivation and construction," in *Proc. Inform. Theory Workshop*, (Cairo, Egypt), Jan. 2010.

[19] S. ten Brink, "Convergence behavior of iteratively decoded parallel concatenated codes," *IEEE Trans. Commun.*, vol. 49, pp. 1727–1737, Oct. 2001.