A GAMP-Based Low Complexity Sparse Bayesian Learning Algorithm

Maher Al-Shoukairi, *Student Member, IEEE*, Philip Schniter, *Fellow, IEEE*, and Bhaskar D. Rao, *Fellow, IEEE*

***Abstract*—In this paper, we present an algorithm for the sparse signal recovery problem that incorporates damped Gaussian generalized approximate message passing (GGAMP) into expectation-maximization-based sparse Bayesian learning (SBL). In particular, GGAMP is used to implement the E-step in SBL in place of matrix inversion, leveraging the fact that GGAMP is guaranteed to converge with appropriate damping. The resulting GGAMP-SBL algorithm is much more robust to arbitrary measurement matrix $\boldsymbol{A}$ than the standard damped GAMP algorithm while being much lower complexity than the standard SBL algorithm. We then extend the approach from the single measurement vector case to the temporally correlated multiple measurement vector case, leading to the GGAMP-TSBL algorithm. We verify the robustness and computational advantages of the proposed algorithms through numerical experiments.**

***Index Terms*—Compressed sensing, approximate message passing (AMP), sparse Bayesian learning (SBL), expectation-maximization algorithms, Gaussian scale mixture, multiple measurement vectors (MMV).**

I. Introduction

A. Sparse Signal Recovery

THE problem of sparse signal recovery (SSR) and the related problem of compressed sensing have received much attention in recent years [1]–[6]. The SSR problem, in the single measurement vector (SMV) case, consists of recovering a sparse signal $\boldsymbol{x} \in \mathbb{R}^N$ from $M \leq N$ noisy linear measurements $\boldsymbol{y} \in \mathbb{R}^M$:

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

where $\boldsymbol{A} \in \mathbb{R}^{M \times N}$ is a known measurement matrix and $\boldsymbol{e} \in \mathbb{R}^M$ is additive noise modeled by $\boldsymbol{e} \sim \mathcal{N}(0, \sigma^2 \boldsymbol{I})$. Despite the difficulty in solving this problem [7], an important finding in recent years is that for a sufficiently sparse $\boldsymbol{x}$ and a well designed $\boldsymbol{A}$, accurate recovery is possible by techniques such as basis pursuit and orthogonal matching pursuit [8]–[10]. The SSR problem has seen considerable advances on the algorithmic front and they include iteratively reweighted algorithms [11]–[13] and Bayesian techniques [14]–[20], among others. Two Bayesian techniques related to this work are the generalized approximate message passing (GAMP) and the sparse Bayesian learning (SBL) algorithms. We briefly discuss both algorithms and some of their shortcomings that we intend to address in this work.

B. Generalized Approximate Message Passing Algorithm

Approximate message passing (AMP) algorithms apply quadratic and Taylor series approximations to loopy belief propagation to produce low complexity algorithms. Based on the original AMP work in [21], a generalized AMP (GAMP) algorithm was proposed in [22]. The GAMP algorithm provides an iterative Bayesian framework under which the knowledge of the matrix $\boldsymbol{A}$ and the densities $p(\boldsymbol{x})$ and $p(\boldsymbol{y}|\boldsymbol{x})$ can be used to compute the maximum a posteriori (MAP) estimate $\hat{\boldsymbol{x}}_{MAP} = \arg\min_{\boldsymbol{x} \in \mathbb{R}^N} p(\boldsymbol{x}|\boldsymbol{y})$ when it is used in its max-sum mode, or approximate the minimum mean-squared error (MMSE) estimate $\hat{\boldsymbol{x}}_{MMSE} = \int_{\mathbb{R}^N} \boldsymbol{x} p(\boldsymbol{x}|\boldsymbol{y}) d\boldsymbol{x} = \mathbb{E}(\boldsymbol{x}|\boldsymbol{y})$ when it is used in its sum-product mode.

The performance of AMP/GAMP algorithms in the large system limit ($M, N \to \infty$) under an i.i.d zero-mean sub-Gaussian matrix $\boldsymbol{A}$ is characterized by state evolution [23], whose fixed points, when unique, coincide with the MAP or the MMSE estimate. However, when $\boldsymbol{A}$ is generic, GAMP's fixed points can be strongly suboptimal and/or the algorithm may never reach its fixed points. Previous work has shown that even mild ill-conditioning or small mean perturbations in $\boldsymbol{A}$ can cause GAMP to diverge [24]–[26]. To overcome the convergence problem in AMP algorithms, a number of AMP modifications have been proposed. A "swept" GAMP (SwAMP) algorithm was proposed in [27], which replaces parallel variable updates in the GAMP algorithm with serial ones to enhance convergence. But SwAMP is relatively slow and still diverges for certain $\boldsymbol{A}$. An adaptive damping and mean-removal procedure for GAMP was proposed in [26] but it too is somewhat slow and still diverges for certain $\boldsymbol{A}$. An alternating direction method of multipliers (ADMM) version of AMP was proposed in [28] with improved robustness but even slower convergence.

In the special case that the prior and likelihood are both independent Gaussian, [24] was able to provide a full characterization of GAMP's convergence. In particular, it was shown that Gaussian GAMP (GGAMP) algorithm will converge if and only if the peak to average ratio of the squared singular values in $\boldsymbol{A}$ is sufficiently small. When this condition is not met, [24]

Manuscript received March 2, 2017; revised June 18, 2017 and August 29, 2017; accepted October 2, 2017. Date of publication October 19, 2017; date of current version December 4, 2017. The associate editor coordinating the review of this manuscript and approving it for publication was Prof. Yimin D. Zhang. The work of M. Al-Shoukairi and B. D. Rao was supported by the Ericsson Endowed Chair Funds. The work of P. Schniter was supported by the National Science Foundation Grant 1527162. *(Corresponding author: Maher Al-Shoukairi.)*

M. Al-Shoukairi and B. D. Rao are with the Department of Electrical and Computer Engineering, University of California at San Diego, San Diego, CA 92093 USA (e-mail: malshouk@ucsd.edu; brao@ucsd.edu).

P. Schniter is with the Department of Electrical and Computer Engineering, The Ohio State University, Columbus, OH 43210 USA (e-mail: schniter@ece.osu.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.2017.2764855

1053-587X © 2017 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission.
See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

proposed a damping technique that guarantees convergence of GGAMP at the expense of slowing its convergence rate. Although the case of Gaussian prior and likelihood is not enough to handle the sparse signal recovery problem directly, it is sufficient to replace the costly matrix inversion step of the standard EM-based implementation of SBL, as we describe below.

C. Sparse Bayesian Learning Algorithm

To understand the contribution of this paper, we give a very brief description of SBL [14], [15], saving a more detailed description for Section II. Essentially, SBL is based on a Gaussian scale mixture (GSM) [29]–[31] prior on $\boldsymbol{x}$. That is, the prior is Gaussian conditioned on a variance vector $\boldsymbol{\gamma}$, which is then controlled by a suitable choice of hyperprior $p(\boldsymbol{\gamma})$. A large of number of sparsity-promoting priors, like the Student-t and Laplacian priors, can be modeled using a GSM, making the approach widely applicable [29]–[32]. In the SBL algorithm, the expectation-maximization (EM) algorithm is used to alternate between estimating $\boldsymbol{\gamma}$ and estimating the signal $\boldsymbol{x}$ under fixed $\boldsymbol{\gamma}$. Since the latter step uses a Gaussian likelihood and Gaussian prior, the exact solution can be computed in closed form via matrix inversion. This matrix inversion is computationally expensive, limiting the algorithms applicability to large scale problems.

D. Paper's Contribution[1]

In this paper, we develop low-complexity algorithms for sparse Bayesian learning (SBL) [14], [15]. Since the traditional implementation of SBL uses matrix inversions at each iteration, its complexity is too high for large-scale problems. In this paper we circumvent the matrix inverse using the generalized approximate message passing (GAMP) algorithm [21], [22], [24]. Using GAMP to implement the E step of EM-based SBL provides a significant reduction in complexity over the classical SBL algorithm. This work is a beneficiary of the algorithmic improvements and theoretical insights that have taken place in recent work in AMP [21], [22], [24], where we exploit the fact that using a Gaussian prior on $p(\boldsymbol{x})$ can provide guarantees for the GAMP E-step to not diverge when sufficient damping is used [24], even for a non-i.i.d.-Gaussian $\boldsymbol{A}$. In other words, the enhanced robustness of the proposed algorithm is due to the GSM prior used on $\boldsymbol{x}$, as opposed to other sparsity promoting priors for which there are no GAMP convergence guarantees when $\boldsymbol{A}$ is non-i.i.d.-Gaussian. The resulting algorithm is the Gaussian GAMP SBL (GGAMP-SBL) algorithm, which combines the robustness of SBL with the speed of GAMP.

To further illustrate and expose the synergy between the AMP and SBL frameworks, we also propose a new approach to the multiple measurement vector (MMV) problem. The MMV problem extends the SMV problem from a single measurement and signal vector to a sequence of measurement and signal vectors. Applications of MMV include direction of arrival (DOA) estimation and EEG/MEG source localization, among others. In our treatment of the MMV problem, all signal vectors are assumed to share the same support. In practice it is often the case that the non-zero signal elements will experience temporal correlation, i.e., each non-zero row of the signal matrix can be treated as a correlated time series. If this correlation is not taken into consideration, the performance of MMV algorithms can degrade quickly [34]. Extensions of SBL to the MMV problem have been developed in [34]–[36], such as the time varying sparse Bayesian learning (TSBL) and TMSBL algorithms [34]. Although TMSBL has lower complexity than TSBL, it still requires an order of $\mathcal{O}(NM^2)$ operations per iteration, making it unsuitable for large-scale problems. To overcome the complexity problem, [37] and [33] proposed AMP-based Bayesian approaches to the MMV problem. However, similar to the SMV case, these algorithms are only expected to work for i.i.d zero-mean sub-Gaussian $\boldsymbol{A}$. We therefore extend the proposed GGAMP-SBL to the MMV case, to produce a GGAMP-TSBL algorithm that is more robust to generic $\boldsymbol{A}$, while achieving linear complexity in all problem dimensions.

The organization of the paper is as follows. In Section II, we review SBL. In Section III, we combine the damped GGAMP algorithm with the SBL approach to solve the SMV problem and investigate its convergence behavior. In Section IV, we use a time-correlated multiple measurement factor graph to derive the GGAMP-TSBL algorithm. In Section V, we present numerical results to compare the performance and complexity of the proposed algorithms with the original SBL and with other AMP algorithms for the SMV case, and with TMSBL for the MMV case. The results show that the new algorithms maintained the robustness of the original SBL and TMSBL algorithms, while significantly reducing their complexity.

II. Sparse Bayesian Learning for SSR

A. GSM Class of Priors

We will assume that the entries of $\boldsymbol{x}$ are independent and identically distributed, i.e. $p(\boldsymbol{x}) = \Pi_n p(x_n)$. The sparsity promoting prior $p(x_n)$ will be chosen from the GSM class and so will admit the following representation

$$p(x_n) = \int \mathcal{N}(x_n; 0, \gamma_n) p(\gamma_n) d\gamma_n, \tag{2}$$

where $\mathcal{N}(x_n; 0, \gamma_n)$ denotes a Gaussian density with mean zero and variance γ_n. The mixing density on hyperprior $p(\gamma_n)$ controls the prior on x_n. For instance, if a Laplacian prior is desired for x_n, then an exponential density is chosen for $p(\gamma_n)$ [29].

In the empirical Bayesian approach, an estimate of the hyperparameter vector $\boldsymbol{\gamma}$ is iteratively estimated, often using evidence maximization. For a given estimate $\hat{\boldsymbol{\gamma}}$, the posterior $p(\boldsymbol{x}|\boldsymbol{y})$ is approximated as $p(\boldsymbol{x}|\boldsymbol{y}; \hat{\boldsymbol{\gamma}})$, and the mean of this posterior is used as a point estimate for $\hat{\boldsymbol{x}}$. This mean can be computed in closed form, as detailed below, because the approximate posterior is Gaussian. It was shown in [38] that this empirical Bayesian method, also referred to as a Type II maximum likelihood method, can be used to formulate a number of algorithms for solving the SSR problem by changing the mixing density $p(\gamma_n)$. There are many computational methods that can be employed for computing $\boldsymbol{\gamma}$ in the evidence maximization framework, e.g, [14], [15], [39]. In this work, we utilize the EM-SBL algorithm because of its synergy with the GAMP framework, as will be apparent below.

[1]Part of this work was presented in the 2014 Asilomar conference on Signals, Systems and Computers [33]

B. SBL's EM Algorithm

In EM-SBL, the EM algorithm is used to learn the unknown signal variance vector $\boldsymbol{\gamma}$ [40]–[42], and possibly also the noise variance σ^2. We focus on learning $\boldsymbol{\gamma}$, assuming the noise variance σ^2 is known. We later state the EM update rule for the noise variance σ^2 for completeness.

The goal of the EM algorithm is to maximize the posterior $p(\boldsymbol{\gamma}|\boldsymbol{y})$ or equivalently[2] to minimize $-\log p(\boldsymbol{y},\boldsymbol{\gamma})$. For the GSM prior (2) and the additive white Gaussian noise model, this results in the SBL cost function [14], [15],

$$\begin{aligned}\chi(\boldsymbol{\gamma}) &= -\log p(\boldsymbol{y},\boldsymbol{\gamma}) \\ &= \frac{1}{2}\log|\boldsymbol{\Sigma}_{\boldsymbol{y}}| + \frac{1}{2}\boldsymbol{y}^\top\boldsymbol{\Sigma}_{\boldsymbol{y}}^{-1}\boldsymbol{y} - \log p(\boldsymbol{\gamma}), \qquad (3)\\ \boldsymbol{\Sigma}_{\boldsymbol{y}} &= \sigma^2\boldsymbol{I} + \boldsymbol{A}\boldsymbol{\Gamma}\boldsymbol{A}^\top,\ \ \boldsymbol{\Gamma} \triangleq \mathrm{Diag}(\boldsymbol{\gamma}).\end{aligned}$$

In the EM-SBL approach, $\boldsymbol{x}$ is treated as the hidden variable and the parameter estimate is iteratively updated as follows:

$$\boldsymbol{\gamma}^{i+1} = \underset{\boldsymbol{\gamma}}{\mathrm{argmax}}\, \mathbb{E}_{\boldsymbol{x}|\boldsymbol{y};\boldsymbol{\gamma}^i}\left[\log p(\boldsymbol{y},\boldsymbol{x},\boldsymbol{\gamma})\right], \qquad (4)$$

where $p(\boldsymbol{y},\boldsymbol{x},\boldsymbol{\gamma})$ is the joint probability of the complete data and $p(\boldsymbol{x}|\boldsymbol{y};\boldsymbol{\gamma}^i)$ is the posterior under the old parameter estimate $\boldsymbol{\gamma}^i$, which is used to evaluate the expectation. In each iteration, an expectation has to be computed (E-step) followed by a maximization step (M-step). It is easy to show that at each iteration, the EM algorithm increases a lower bound on the log posterior $\log p(\boldsymbol{\gamma}|\boldsymbol{y})$ [40], and it has been shown in [42] that the algorithm will converge to a stationary point of the posterior under a fairly general set of conditions that are applicable in many practical applications.

Next we detail the implementation of the E and M steps of the EM-SBL algorithm.

SBL's E-step: The Gaussian assumption on the additive noise $\boldsymbol{e}$ leads to the following Gaussian likelihood function:

$$p(\boldsymbol{y}|\boldsymbol{x};\sigma^2) = \frac{1}{(2\pi\sigma^2)^{\frac{M}{2}}}\exp\left(-\frac{1}{2\sigma^2}\|\boldsymbol{y}-\boldsymbol{A}\boldsymbol{x}\|^2\right). \qquad (5)$$

Due to the GSM prior (2), the density of $\boldsymbol{x}$ conditioned on $\boldsymbol{\gamma}$ is Gaussian:

$$p(\boldsymbol{x}|\boldsymbol{\gamma}) = \prod_{n=1}^{N}\frac{1}{(2\pi\gamma_n)^{\frac{1}{2}}}\exp\left(-\frac{x_n^2}{2\gamma_n}\right). \qquad (6)$$

Putting (5) and (6) together, the density needed for the E-step is Gaussian:

$$\begin{aligned}p(\boldsymbol{x}|\boldsymbol{y},\boldsymbol{\gamma}) &= \mathcal{N}(\boldsymbol{x};\hat{\boldsymbol{x}},\boldsymbol{\Sigma}_{\boldsymbol{x}}) \qquad (7)\\ \hat{\boldsymbol{x}} &= \sigma^{-2}\boldsymbol{\Sigma}_{\boldsymbol{x}}\boldsymbol{A}^\top\boldsymbol{y} \qquad (8)\\ \boldsymbol{\Sigma}_{\boldsymbol{x}} &= (\sigma^{-2}\boldsymbol{A}^\top\boldsymbol{A} + \boldsymbol{\Gamma}^{-1})^{-1} \\ &= \boldsymbol{\Gamma} - \boldsymbol{\Gamma}\boldsymbol{A}^\top(\sigma^2\boldsymbol{I} + \boldsymbol{A}\boldsymbol{\Gamma}\boldsymbol{A}^\top)^{-1}\boldsymbol{A}\boldsymbol{\Gamma}. \qquad (9)\end{aligned}$$

We refer to the mean vector as $\hat{\boldsymbol{x}}$ since it will be used as the SBL point estimate of $\boldsymbol{x}$. In the sequel, we will use $\boldsymbol{\tau}_x$ when referring to the vector composed from the diagonal entries of the covariance matrix $\boldsymbol{\Sigma}_{\boldsymbol{x}}$. Although both $\hat{\boldsymbol{x}}$ and $\boldsymbol{\tau}_x$ change with the iteration i, we will sometimes omit their i dependence for brevity. Note that the mean and covariance computations in (8) and (9) are not affected by the choice of $p(\boldsymbol{\gamma})$. The mean and diagonal entries of the covariance matrix are needed to carry out the M-step as shown next.

SBL's M-Step: The M-step is then carried out as follows. First notice that

$$\begin{aligned}&\mathbb{E}_{\boldsymbol{x}|\boldsymbol{y};\boldsymbol{\gamma}^i,\sigma^2}\left[-\log p(\boldsymbol{y},\boldsymbol{x},\boldsymbol{\gamma};\sigma^2)\right] = \\ &\mathbb{E}_{\boldsymbol{x}|\boldsymbol{y};\boldsymbol{\gamma}^i,\sigma^2}\left[-\log p(\boldsymbol{y}|\boldsymbol{x};\sigma^2) - \log p(\boldsymbol{x}|\boldsymbol{\gamma}) - \log p(\boldsymbol{\gamma})\right]. \qquad (10)\end{aligned}$$

Since the first term in (10) does not depend on $\boldsymbol{\gamma}$, it will not be relevant for the M-step and thus can be ignored. Similarly, in the subsequent steps we will drop constants and terms that do not depend on $\boldsymbol{\gamma}$ and therefore do not impact the M-step. Since $\mathbb{E}_{\boldsymbol{x}|\boldsymbol{y};\boldsymbol{\gamma}^i,\sigma^2}[x_n^2] = \hat{x}_n^2 + \tau_{x_n}$,

$$\begin{aligned}&\mathbb{E}_{\boldsymbol{x}|\boldsymbol{y};\boldsymbol{\gamma}^i,\sigma^2}\left[-\log p(\boldsymbol{x}|\boldsymbol{\gamma}) - \log p(\boldsymbol{\gamma})\right] = \\ &\sum_{n=1}^{N}\left(\left(\frac{\hat{x}_n^2+\tau_{x_n}}{2\gamma_n}\right) + \frac{1}{2}\log\gamma_n - \log p(\gamma_n)\right). \qquad (11)\end{aligned}$$

Note that the E-step only requires $\hat{x}_n$, the posterior mean from (8), and τ_{x_n} ,the posterior variance from (9), which are statistics of the marginal densities $p(x_n|\boldsymbol{y},\boldsymbol{\gamma}^i)$. In other words, the full joint posterior $p(\boldsymbol{x}|\boldsymbol{y},\boldsymbol{\gamma}^i)$ is not needed. This facilitates the use of message passing algorithms.

As can be seen from (7)–(9), the computation of $\hat{\boldsymbol{x}}$ and $\boldsymbol{\tau}_x$ involves the inversion of an $N \times N$ matrix, which can be reduced to $M \times M$ matrix inversion by the matrix inversion lemma. The complexity of computing $\hat{\boldsymbol{x}}$ and $\boldsymbol{\tau}_x$ can be shown to be $\mathcal{O}(NM^2)$ under the assumption that $M \leq N$. This makes the EM-SBL algorithm computationally prohibitive and impractical to use with large dimensions.

From (4) and (11), the M-step for each iteration is as follows:

$$\boldsymbol{\gamma}^{i+1} = \underset{\boldsymbol{\gamma}}{\mathrm{argmin}}\left[\sum_{n=1}^{N}\left(\frac{\hat{x}_n^2+\tau_{x_n}}{2\gamma_n} + \frac{\log\gamma_n}{2} - \log p(\gamma_n)\right)\right]. \qquad (12a)$$

This reduces to *N* scalar optimization problems,

$$\gamma_n^{i+1} = \underset{\gamma_n}{\mathrm{argmin}}\left[\frac{\hat{x}_n^2+\tau_{x_n}}{2\gamma_n} + \frac{1}{2}\log\gamma_n - \log p(\gamma_n)\right]. \qquad (12b)$$

The choice of hyperprior $p(\boldsymbol{\gamma})$ plays a role in the M-step, and governs the prior for $\boldsymbol{x}$. However, from the computational simplicity of the M-step, as evident from (12b), the hyperprior rarely impacts the overall algorithmic computational complexity, which is mainly that of computing the quantities $\hat{\boldsymbol{x}}$ and $\boldsymbol{\tau}_x$ in the E-step.

Often a non-informative prior is used in SBL. For the purpose of obtaining the M-step update, we will also simplify and drop $p(\boldsymbol{\gamma})$ and compute the Maximum Likelihood estimate of $\boldsymbol{\gamma}$. From (12b), this reduces to, $\gamma_n^{i+1} = \hat{x}_n^2 + \tau_{x_n}$.

[2]Using Bayes rule, $p(\boldsymbol{\gamma}|\boldsymbol{y}) = p(\boldsymbol{y},\boldsymbol{\gamma})/p(\boldsymbol{y})$ where $p(\boldsymbol{y})$ is a constant with respect to $\boldsymbol{\gamma}$. Thus for MAP estimation of $\boldsymbol{\gamma}$ we can maximize $p(\boldsymbol{y},\boldsymbol{\gamma})$, or minimize $-\log p(\boldsymbol{y},\boldsymbol{\gamma})$.

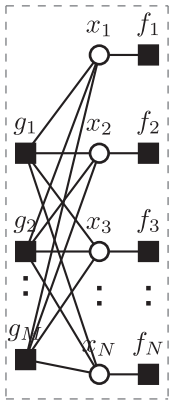


Fig. 1. GAMP Factor Graph.

Similarly, if the noise variance σ^2 is unknown, it can be estimated using:

$$(\sigma^2)^{i+1} = \underset{\sigma^2}{\operatorname{argmax}}\, \mathbb{E}_{\boldsymbol{x}|\boldsymbol{y},\boldsymbol{\gamma};(\sigma^2)^i}[p(\boldsymbol{y},\boldsymbol{x},\boldsymbol{\gamma};\sigma^2)]$$
$$= \frac{\|\boldsymbol{y}-\boldsymbol{A}\boldsymbol{x}\|^2 + (\sigma^2)^i \sum_{n=1}^{N}\left(1-\frac{\tau_{x_n}}{\gamma_n}\right)}{M}. \quad (13)$$

We note here that estimates obtained by (13) can be highly inaccurate as mentioned in [35]. Therefore, it suggests that experimenting with different values of σ^2 or using some other application based heuristic will probably lead to better results.

III. Damped Gaussian GAMP SBL

We now show how damped GGAMP can be used to simplify the E-step above, leading to the damped GGAMP-SBL algorithm. Then we examine the convergence behavior of the resulting algorithm.

A. GGAMP-SBL

Above we showed that, in the EM-SBL algorithm, the M-step is computationally simple but the E-step is computationally demanding. The GAMP algorithm can be used to efficiently approximate the quantities $\hat{\boldsymbol{x}}$ and $\boldsymbol{\tau}_x$ needed in the E-step, while the M-step remains unchanged. GAMP is based on the factor graph in Fig. 1, where for a given prior $f_n(x) = p(x_n)$ and a likelihood function $g_m = p(y_m|\boldsymbol{x})$, GAMP uses quadratic approximations and Taylor series expansions, to provide approximations of MAP or MMSE estimates of $\boldsymbol{x}$. The reader can refer to [22] for detailed derivation of GAMP. The E-step in Table I, uses the damped GGAMP algorithm from [24] because of its ability to enhance traditional GAMP algorithm divergence issues with non-i.i.d.-Gaussian $\boldsymbol{A}$. The damped GGAMP algorithm has an important modification over the original GAMP algorithm and also over the previously proposed AMP-SBL [33], namely the introduction of damping factors $\theta_s, \theta_x \in (0,1]$ to slow down updates and enhance convergence. Setting $\theta_s = \theta_x = 1$ in the damped GGAMP algorithm will yield no damping, and reduces the algorithm to the original GAMP algorithm. We note here that the damped GGAMP algorithm from [24] is referred to by GGAMP, and therefore we will be using the terms GGAMP and damped GGAMP interchangeably in this paper. Moreover, when

TABLE I
GGAMP-SBL Algorithm

Initialization	
$\boldsymbol{S} \leftarrow \lvert\boldsymbol{A}\rvert^2$ (component wise magnitude squared)	(I1)
Initialize $\breve{\boldsymbol{\tau}}_x^0, \boldsymbol{\gamma}^0, (\sigma^2)^0 > 0$	(I2)
$\breve{\boldsymbol{s}}^0, \breve{\boldsymbol{x}}^0 \leftarrow \boldsymbol{0}$	(I3)
for $i = 1, 2, ..., I_{\max}$	
Initialize $\boldsymbol{\tau}_x^1 \leftarrow \breve{\boldsymbol{\tau}}_x^{i-1}, \hat{\boldsymbol{x}}^1 \leftarrow \breve{\boldsymbol{x}}^{i-1}, \boldsymbol{s}^1 \leftarrow \breve{\boldsymbol{s}}^{i-1}$	
E-Step approximation	
for $k = 1, 2, ..., K_{\max}$	
$1/\boldsymbol{\tau}_p^k \leftarrow \boldsymbol{S}\boldsymbol{\tau}_x^k$	(A1)
$\boldsymbol{p}^k \leftarrow \boldsymbol{s}^{k-1} + \boldsymbol{\tau}_p^k \boldsymbol{A}\hat{\boldsymbol{x}}^k$	(A2)
$\boldsymbol{\tau}_s^k \leftarrow \boldsymbol{\tau}_p^k g_s'(\boldsymbol{p}^k, \boldsymbol{\tau}_p^k)$	(A3)
$\boldsymbol{s}^k \leftarrow (1-\theta_s)\boldsymbol{s}^{k-1} + \theta_s g_s(\boldsymbol{p}^k, \boldsymbol{\tau}_p^k)$	(A4)
$1/\boldsymbol{\tau}_r^k \leftarrow \boldsymbol{S}^\top \boldsymbol{\tau}_s^k$	(A5)
$\boldsymbol{r}^k \leftarrow \hat{\boldsymbol{x}}^k - \boldsymbol{\tau}_r^k \boldsymbol{A}^\top \boldsymbol{s}^k$	(A6)
$\boldsymbol{\tau}_x^{k+1} \leftarrow \boldsymbol{\tau}_r^k g_x'(\boldsymbol{r}^k, \boldsymbol{\tau}_r^k)$	(A7)
$\hat{\boldsymbol{x}}^{k+1} \leftarrow (1-\theta_x)\hat{\boldsymbol{x}}^k + \theta_x g_x(\boldsymbol{r}^k, \boldsymbol{\tau}_r^k)$	(A8)
if $\|\hat{\boldsymbol{x}}^{k+1} - \hat{\boldsymbol{x}}^k\|^2 / \|\hat{\boldsymbol{x}}^{k+1}\|^2 < \epsilon_{\text{gamp}}$, break	(A9)
end for %end of k loop	
$\breve{\boldsymbol{s}}^i \leftarrow \boldsymbol{s}^k$, $\breve{\boldsymbol{x}}^i \leftarrow \hat{\boldsymbol{x}}^{k+1}$, $\breve{\boldsymbol{\tau}}_x^i \leftarrow \boldsymbol{\tau}_x^{k+1}$	
M-Step	
$\boldsymbol{\gamma}^{i+1} \leftarrow \lvert\breve{\boldsymbol{x}}^i\rvert^2 + \breve{\boldsymbol{\tau}}_x^i$	(M1)
$(\sigma^2)^{i+1} \leftarrow \frac{\|\boldsymbol{y}-\boldsymbol{A}\breve{\boldsymbol{x}}^i\|^2 + (\sigma^2)^i \sum_{n=1}^{N}\left(1-\frac{\breve{\tau}_{x_n}^i}{\gamma_n^i}\right)}{M}$	(M2)
if $\|\breve{\boldsymbol{x}}^i - \breve{\boldsymbol{x}}^{i-1}\|^2 / \|\breve{\boldsymbol{x}}^i\|^2 < \epsilon_{\text{em}}$, break	(M3)
end for %end of i loop	

the components of the matrix $\boldsymbol{A}$ are not zero-mean, one can incorporate the same mean removal technique used in [26]. The input and output functions $g_s(\boldsymbol{p}, \boldsymbol{\tau}_p)$ and $g_x(\boldsymbol{r}, \boldsymbol{\tau}_r)$ in Table I are defined based on whether the max-sum or the sum-product version of GAMP is being used. The intermediate variables $\boldsymbol{r}$ and $\boldsymbol{p}$ are interpreted as approximations of Gaussian noise corrupted versions of $\boldsymbol{x}$ and $\boldsymbol{z} = \boldsymbol{A}\boldsymbol{x}$, with the respective noise levels of $\boldsymbol{\tau}_r$ and $\boldsymbol{\tau}_p$. In the max-sum version, the vector MAP estimation problem is reduced to a sequence of scalar MAP estimates given $\boldsymbol{r}$ and $\boldsymbol{p}$ using the input and output functions, where they are defined as:

$$[g_s(\boldsymbol{p}, \boldsymbol{\tau}_p)]_m = p_m - \tau_{p_m} \operatorname{prox}_{\frac{-1}{\tau_{p_m}} \ln p(y_m|z_m)}\left(\frac{p_m}{\tau_{p_m}}\right) \quad (14)$$

$$[g_x(\boldsymbol{r}, \boldsymbol{\tau}_r)]_n = \operatorname{prox}_{-\tau_{r_n} \ln p(x_n)}(r_n) \quad (15)$$

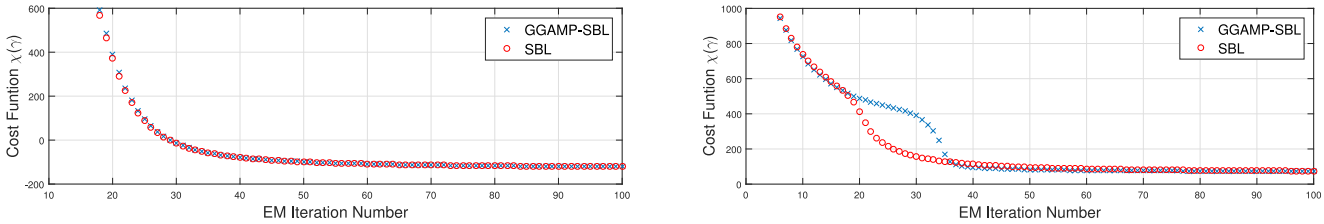
$$\operatorname{prox}_f(r) \triangleq \underset{x}{\operatorname{argmin}}\, f(x) + \frac{1}{2}|x-r|^2. \quad (16)$$

Similarly, in the sum-product version of the algorithm, the vector MMSE estimation problem is reduced to a sequence of scalar MMSE estimates given $\boldsymbol{r}$ and $\boldsymbol{p}$ using the input and output functions, where they are defined as:

$$[g_s(\boldsymbol{p}, \boldsymbol{\tau}_p)]_m = \frac{\int z_m p(y_m|z_m)\mathcal{N}(z_m; \frac{p_m}{\tau_{p_m}}, \frac{1}{\tau_{p_m}})dz_m}{\int p(y_m|z_m)\mathcal{N}(z_m; \frac{p_m}{\tau_{p_m}}, \frac{1}{\tau_{p_m}})dz_m} \quad (17)$$

$$[g_x(\boldsymbol{r}, \boldsymbol{\tau}_r)]_n = \frac{\int x_n p(x_n)\mathcal{N}(x_n; r_n, \tau_{r_n})dx_n}{\int p(x_n)\mathcal{N}(x_n; r_n, \tau_{r_n})dx_n}. \quad (18)$$

For the parametrized Gaussian prior we imposed on $\boldsymbol{x}$ in (2), both sum-product and max-sum versions of $g_x(\boldsymbol{r}, \boldsymbol{\tau}_r)$ yield the



(a) Cost functions for i.i.d.-Gaussian A

(b) Cost functions for column correlated A

Fig. 2. Cost functions on SBL and GGAMP-SBL algorithms versus number of EM iterations. (a) Cost functions for i.i.d.-Gaussian A. (b) Cost functions for column correlated A.

same updates for $\hat{\boldsymbol{x}}$ and $\boldsymbol{\tau}_x$ [22], [24]:

$$g_x(\boldsymbol{r}, \boldsymbol{\tau}_r) = \frac{\gamma}{\gamma + \boldsymbol{\tau}_{\boldsymbol{r}}} \boldsymbol{r} \tag{19}$$

$$g_x'(\boldsymbol{r}, \boldsymbol{\tau}_r) = \frac{\gamma}{\gamma + \boldsymbol{\tau}_r}. \tag{20}$$

Similarly, in the case of the likelihood $p(\boldsymbol{y}|\boldsymbol{x})$ given in (5), the max-sum and sum-product versions of $g_s(\boldsymbol{p}, \boldsymbol{\tau}_p)$ yield the same updates for $\boldsymbol{s}$ and $\boldsymbol{\tau}_s$ [22], [24]:

$$g_s(\boldsymbol{p}, \boldsymbol{\tau}_p) = \frac{(\boldsymbol{p}/\boldsymbol{\tau}_p - \boldsymbol{y})}{(\sigma^2 + 1/\boldsymbol{\tau}_p)} \tag{21}$$

$$g_s'(\boldsymbol{p}, \boldsymbol{\tau}_p) = \frac{\sigma^{-2}}{\sigma^{-2} + \boldsymbol{\tau}_p}. \tag{22}$$

We note that, in (19), (20), (21) and (22), and for all equations in Table I, all vector squares, divisions and multiplications are taken element wise.

In Table I, $K_{\max}$ is the maximum allowed number of GAMP algorithm iterations, ϵ_{gamp} is the GAMP normalized tolerance parameter, $I_{\max}$ is the maximum allowed number of EM iterations and ϵ_{em} is the EM normalized tolerance parameter. Upon the convergence of GAMP algorithm based E-step, estimates for the mean $\hat{\boldsymbol{x}}$ and covariance diagonal $\boldsymbol{\tau}_x$ are obtained. These estimates can be used in the M-step of the algorithm, given by (12b). These estimates, along with the $\boldsymbol{s}$ vector estimate, are also used to initialize the E-step at the next EM iteration to accelerate the convergence of the overall algorithm.

Defining $\boldsymbol{S}$ as the component wise magnitude squared of $\boldsymbol{A}$, the complexity of the GGAMP-SBL algorithm is dominated by the E-step, which in turn (from Table I) is dominated by the matrix multiplications by $\boldsymbol{A}$, $\boldsymbol{A}^\top$, $\boldsymbol{S}$ and $\boldsymbol{S}^\top$ at each iteration, implying that the computational cost of the algorithm is $\mathcal{O}(NM)$ operations per GAMP algorithm iteration multiplied by the total number of GAMP algorithm iterations. For large M, this is much smaller than $\mathcal{O}(NM^2)$, the complexity of standard SBL iteration.

In addition to the complexity of each iteration, for the proposed GGAMP-SBL algorithm to achieve faster runtimes it is important for GGAMP-SBL total number of iterations to not be too large, to the point where it over weighs the reduction in complexity per iteration, especially when heavier damping is used. We point out here that while SBL provides a one step exact solution for the E-step, GGAMP-SBL provides an approximate iterative solution. Based on that, the total number of SBL iterations is the number of EM iterations needed for convergence, while the total number of GGAMP-SBL iterations is based on the number of EM iterations it needs to converge and the number of E-step iterations for each EM iteration. First we consider the number of EM iterations for both algorithms. As explained in Section III-B2, the E-step of GGAMP-SBL algorithm provides a good approximation of the true posterior [43]. In addition to that the number of EM iterations is not affected by damping, since damping only affects the number of iterations of GGAMP in the E-step, but it does not affect its outcome upon convergence. Based on these two points, we can expect the number of EM iterations for GGAMP-SBL to be generally in the same range as the original SBL algorithm. This is also shown in Section III-B2 Fig. 2(a) and (b), where we can see the two cost functions being reduced to their minimum values using approximately the same number of EM iterations, even when heavier damping is used. As for the GGAMP-SBL E-step iterations, because we are warm starting each E-step with $\boldsymbol{x}$ and $\boldsymbol{s}$ values from the previous EM iteration, it was found through numerical experiments that the number of required E-step iterations is reduced each time, to the point where the E-step converges to the required tolerance within 2–3 iterations towards the final EM iterations. When averaging the total number of E-step iterations over the number of EM iterations, it was found that for medium to large problem sizes the average number of E-step iterations was just a fraction of the measurements number M, even in the cases where heavier damping was used. Moreover, it was observed that the number of iterations required for the E-step to converge is independent of the problem size, which gives the algorithm a bigger advantage at larger problem sizes. Finally, based on the complexity reduction per iteration and the total number of iterations required for GGAMP-SBL, we can expect it to have lower runtimes than SBL for medium to large problems, even when heavier damping is used. This runtime advantage is confirmed through numerical experiments in Section V.

B. GGAMP-SBL Convergence

We now examine the convergence of the GGAMP-SBL algorithm. This involves two steps; the first step is to show that the approximate message passing algorithm employed in the E-step converges and the second step is to show that the overall EM algorithm, which consists of several E and M-steps, converges. For the second step, in addition to convergence of the E-step (first step), the accuracy of the resulting estimates is important. Therefore, in the second step of our convergence investigation, we use results from [43], in addition to numerical results to show that the GGAMP-SBL's E and M steps are actually descending on the original SBL's cost function (3) at each EM iteration.

1) Convergence of the E-step With Generic Transformations: For the first step, we use the analysis from [24] which shows that, in the case of generic $\boldsymbol{A}$, the damped GGAMP algorithm is guaranteed to globally converge (to some values $\hat{\boldsymbol{x}}$ and $\boldsymbol{\tau}_x$) when

sufficient damping is used. In particular, since $\boldsymbol{\gamma}^i$ is fixed in the E-step, the prior is Gaussian and so based on results in [24], starting with an initial estimate $\boldsymbol{\tau}_x \geq \boldsymbol{\gamma}^i$ the variance updates $\boldsymbol{\tau}_x$, $\boldsymbol{\tau}_s$, $\boldsymbol{\tau}_r$ and $\boldsymbol{\tau}_p$ will converge to a unique fixed point. In addition, any fixed point $(\boldsymbol{s}, \hat{\boldsymbol{x}})$ for GGAMP is globally stable if $\theta_s \theta_x ||\tilde{\boldsymbol{A}}||_2^2 < 1$, where the matrix $\tilde{\boldsymbol{A}}$ is defined as given below and is based on the fixed-point values of $\boldsymbol{\tau}_p$ and $\boldsymbol{\tau}_r$:

$$\tilde{\boldsymbol{A}} := \text{Diag}^{1/2}(\boldsymbol{\tau}_p \boldsymbol{q}_s) \boldsymbol{A} \, \text{Diag}^{1/2}(\boldsymbol{\tau}_r \boldsymbol{q}_x)$$

$$\boldsymbol{q}_s = \frac{\sigma^{-2}}{\sigma^{-2} + \boldsymbol{\tau}_p}, \; \boldsymbol{q}_x = \frac{\boldsymbol{\gamma}}{\boldsymbol{\gamma} + \boldsymbol{\tau}_r}.$$

While the result above establishes that the GGAMP algorithm is guaranteed to converge when sufficient amount of damping is used at each iteration, in practice we do not recommend building the matrix $\tilde{\boldsymbol{A}}$ at each EM iteration and calculating its spectral norm. Rather, we recommend choosing sufficiently small damping factors θ_x and θ_s and fixing them for all GGAMP-SBL iterations. For this purpose, the following result from [24] for an i.i.d.-Gaussian prior $p(\boldsymbol{x}) = \mathcal{N}(\boldsymbol{x}; \boldsymbol{0}, \gamma_x \boldsymbol{I})$ can provide some guidance on choosing the damping factors. For the i.i.d.-Gaussian prior case, the damped GAMP algorithm is shown to converge if

$$\Omega(\theta_s, \theta_x) > \|\boldsymbol{A}\|_2^2 / \|\boldsymbol{A}\|_F^2, \tag{23}$$

where $\Omega(\theta_s, \theta_x)$ is defined as

$$\Omega(\theta_s, \theta_x) := \frac{2[(2 - \theta_x)N + \theta_x M]}{\theta_x \theta_s MN}. \tag{24}$$

Experimentally, it was found that using a threshold $\Omega(\theta_s, \theta_x)$ that is 10% larger than (24) is sufficient for the GGAMP-SBL algorithm to converge in the scenarios we considered.

2) GGAMP-SBL Convergence: The result above guarantees convergence of the E-step to some vectors $\hat{\boldsymbol{x}}$ and $\boldsymbol{\tau}_x$ but it does not provide information about the overall convergence of the EM algorithm to the desired SBL fixed points. This convergence depends on the quality of the mean $\hat{\boldsymbol{x}}$ and variance $\boldsymbol{\tau}_x$ computed by the GGAMP algorithm. It has been shown that for an arbitrary $\boldsymbol{A}$ matrix, the fixed-point value of $\hat{\boldsymbol{x}}$ will equal the true mean given in (8) [43]. As for the variance updates, based on the state evolution in [23], the vector $\boldsymbol{\tau}_x$ will equal the true posterior variance vector, i.e., the diagonal of (9), in the case that $\boldsymbol{A}$ is large and i.i.d. Gaussian, but otherwise will only approximate the true quantity.

The approximation of $\boldsymbol{\tau}_x$ by the GGAMP algorithm in the E-step introduces an approximation in the GGAMP-SBL algorithm compared to the original EM-SBL algorithm. Fortunately, there is some flexibility in the EM algorithm in that the M-step need not be carried out to minimize the objective stated in (12a) but it is sufficient to decrease the objective function as discussed in the generalized EM algorithm literature [41], [42]. Given that the mean is estimated accurately, EM iterations may be tolerant to some error in the variance estimate. Some flexibility in this regards can also be gleaned from the results in [44], where it is shown how different iteratively reweighted algorithms correspond to a different choice in the variance. However, we have not been able to prove rigorously that the GGAMP approximation will guarantee descent of the original cost function given in (3).

Nevertheless, our numerical experiments suggest that the GGAMP approximation has negligible effect on algorithm convergence and ability to recover sparse solutions. We select two experiments to illustrate the convergence behavior and demonstrate that the approximate variance estimates are sufficient to decrease SBL's cost function (3). In both experiments $\boldsymbol{x}$ is drawn from a Bernoulli-Gaussian distribution with a non-zero probability λ set to 0.2, and we set $N = 1000$ and $M = 500$. Fig. 2 shows a comparison between the original SBL and the GGAMP-SBL's cost functions at each EM iteration of the algorithms. $\boldsymbol{A}$ in Fig. 2(a) is i.i.d.-Gaussian, while in Fig. 2(b) it is a column correlated transformation matrix, which is constructed according to the description given in Section V, with correlation coefficient $\rho = 0.9$.

The cost functions in Fig. 2(a) and (b) show that, although we are using an approximate variance estimate to implement the M-step, the updates are decreasing the SBL's cost function at each iteration. As noted previously, it is not necessary for the M-step to provide the maximum cost function reduction, it is sufficient to provide some reduction in the cost function for the EM algorithm to be effective. The cost function plots confirm this principle, since GGAMP-SBL eventually reaches the same minimal value as the original EM-SBL. While the two numerical experiments do not provide a guarantee that the overall GGAMP-SBL algorithm will converge, they suggest that the performance of the GGAMP-SBL algorithm often matches that of the original EM-SBL algorithm, which is supported by the more extensive numerical results in Section V.

IV. GGAMP-TSBL for the MMV Problem

In this section, we apply the damped GAMP algorithm to the MMV empirical Bayesian approach to derive a low complexity algorithm for the MMV case as well. Since the GAMP algorithm was originally derived for the SMV case using an SMV factor graph [22], extending it to the MMV case requires some more effort and requires going back to the factor graphs that are the basis of the GAMP algorithm, making some adjustments, and then utilizing the GAMP algorithm.

Once again we use an empirical Bayesian approach with a GSM model, and we focus on the ML estimate of $\boldsymbol{\gamma}$. We assume a common sparsity profile between all measured vectors, and also account for the temporal correlation that might exist between the non-zero signal elements. Previous Bayesian algorithms that have shown good recovery performance for the MMV problem include extensions of the SMV SBL algorithm, such as MSBL [35], TSBL and TMSBL [34]. MSBL is a straightforward extension of SMV SBL, where no temporal correlation between non-zero elements is assumed, while TSBL and TMSBL account for temporal correlation. Even though the TMSBL algorithm has lower complexity compared to the TSBL algorithm, the algorithm still has complexity of $\mathcal{O}(NM^2)$, which can limit its utility when the problem dimensions are large. Other AMP based Bayesian algorithms have achieved linear complexity in the problem dimensions, like AMP-MMV [37]. However AMP-MMV's robustness to generic $\boldsymbol{A}$ matrices is expected to be outperformed by an SBL based approach.

A. MMV Model and Factor Graph

The MMV model can be stated as:

$$\boldsymbol{y}^{(t)} = \boldsymbol{A}\boldsymbol{x}^{(t)} + \boldsymbol{e}^{(t)}, \;\; t = 1, 2, ..., T,$$

where we have T measurement vectors $[\boldsymbol{y}^{(1)}, \boldsymbol{y}^{(2)} ..., \boldsymbol{y}^{(T)}]$ with $\boldsymbol{y}^{(t)} \in \mathbb{R}^M$. The objective is to recover $\boldsymbol{X} = [\boldsymbol{x}^{(t)}, \boldsymbol{x}^{(2)} ..., \boldsymbol{x}^{(T)}]$ with $\boldsymbol{x}^{(t)} \in \mathbb{R}^N$, where in addition to the vectors $\boldsymbol{x}^{(t)}$ being sparse, they share the same sparsity profile. Similar to the SMV case, $\boldsymbol{A} \in \mathbb{R}^{M \times N}$ is known, and $[\boldsymbol{e}^{(1)}, \boldsymbol{e}^{(2)} ..., \boldsymbol{e}^{(T)}]$ is a sequence of i.i.d. noise vectors modeled as $\boldsymbol{e}^{(t)} \sim \mathcal{N}(0, \sigma^2 \boldsymbol{I})$. This model can be restated as:

$$\bar{\boldsymbol{y}} = \boldsymbol{D}(\boldsymbol{A})\bar{\boldsymbol{x}} + \bar{\boldsymbol{e}},$$

where $\bar{\boldsymbol{y}} \triangleq [\boldsymbol{y}^{(1)^\top}, \boldsymbol{y}^{(2)^\top} ..., \boldsymbol{y}^{(T)^\top}]^\top$, $\bar{\boldsymbol{x}} \triangleq [\boldsymbol{x}^{(1)^\top}, \boldsymbol{x}^{(2)^\top} ..., \boldsymbol{x}^{(T)^\top}]^\top$, $\bar{\boldsymbol{e}} \triangleq [\boldsymbol{e}^{(1)^\top}, \boldsymbol{e}^{(2)^\top} ..., \boldsymbol{e}^{(T)^\top}]^\top$ and $\boldsymbol{D}(\boldsymbol{A})$ is a block-diagonal matrix constructed from T replicas of $\boldsymbol{A}$.

The posterior distribution of $\bar{\boldsymbol{x}}$ is given by:

$$p(\bar{\boldsymbol{x}} \mid \bar{\boldsymbol{y}}) \propto \prod_{t=1}^{T} \left[\prod_{m=1}^{M} p(y_m^{(t)} \mid \boldsymbol{x}^{(t)}) \prod_{n=1}^{N} p(x_n^{(t)} \mid x_n^{(t-1)}) \right],$$

where

$$p(y_m^{(t)} \mid \boldsymbol{x}^{(t)}) = \mathcal{N}(y_m^{(t)}; \boldsymbol{a}_{m,.}^\top \boldsymbol{x}^{(t)}, \sigma^2),$$

where $\boldsymbol{a}_{m,.}^\top$ is the mth row of the matrix $\boldsymbol{A}$. Similar to the previous work in [36], [37], [45], we use an AR(1) process to model the correlation between $x_n^{(t)}$ and $x_n^{(t-1)}$, i.e.,

$$\begin{aligned} x_n^{(t)} &= \beta x_n^{(t-1)} + \sqrt{1-\beta^2} v_n^{(t)} \\ p(x_n^{(t)} \mid x_n^{(t-1)}) &= \mathcal{N}(x_n^{(t)}; \beta x_n^{(t-1)}, (1-\beta^2)\gamma_n), \ t > 1 \\ p(x_n^{(1)}) &= \mathcal{N}(x_n^{(1)}; 0, \gamma_n), \end{aligned}$$

where $\beta \in (-1, 1)$ is the temporal correlation coefficient and $v_n^{(t)} \sim \mathcal{N}(0, \gamma_n)$. Following an empirical Bayesian approach similar to the one proposed for the SMV case, the hyperparameter vector $\boldsymbol{\gamma}$ is then learned from the measurements using the EM algorithm. The EM algorithm can also be used to learn the correlation coefficient β and the noise variance σ^2. Based on these assumptions we use the sum-product algorithm [46] to construct the factor graph in Fig. 3, and derive the MMV algorithm GGAMP-TSBL. In the MMV factor graph, the factors are $g_m^{(t)}(\boldsymbol{x}) = p(y_m^{(t)} \mid \boldsymbol{x}^{(t)})$, $f_n^{(t)}(x_n^{(t)}) = p(x_n^{(t)} \mid x_n^{(t-1)})$ for $t > 1$ and $f_n^{(1)}(x_n^{(1)}) = p(x_n^{(1)})$.

B. GGAMP-TSBL Message Phases and Scheduling (E-Step)

Due to the similarities between the factor graph for each time frame of the MMV model and the factor graph of the SMV model, we will use the algorithm in Table I as a building block and extend it to the MMV case. We divide the message updates into three steps as shown in Fig. 4.

For each time frame the "within" step in Fig. 4 is very similar to the SMV GAMP iteration, with the only difference being that each $x_n^{(t)}$ is connected to the factor nodes $f_n^{(t)}$ and $f_n^{(t+1)}$, while it is connected to one factor node in the SMV case. This difference is reflected in the calculation of the output function

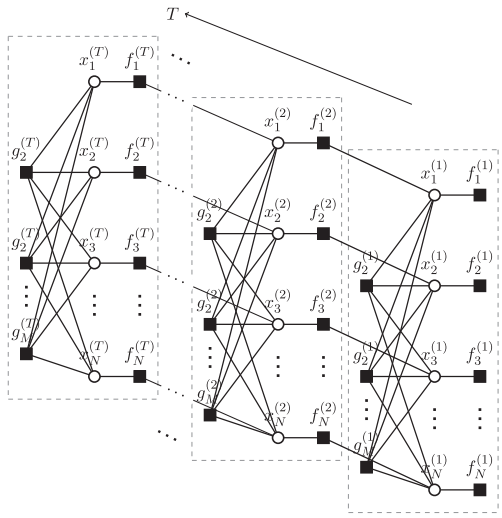


Fig. 3. GGAMP-TSBL factor graph.

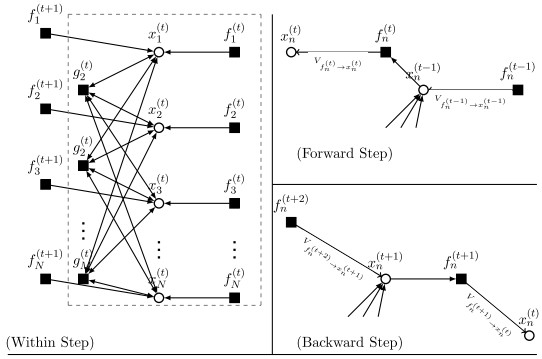


Fig. 4. Message passing phases for GGAMP-TSBL.

g_x and therefore in finding the mean and variance estimates for $\bar{\boldsymbol{x}}$. The details of finding g_x and therefore the update equations for $\boldsymbol{\tau}_x^{(t)}$ and $\hat{\boldsymbol{x}}^{(t)}$ are shown in Appendix A. The input function g_s is the same as (21), and the update equations for $\boldsymbol{\tau}_s^{(t)}$ and $\boldsymbol{s}^{(t)}$ are the same as (A3) and (A4) from Table I, because an AWGN model is assumed for the noise. The second type of updates are passing messages forward in time from $x_n^{(t-1)}$ to $x_n^{(t)}$ through $f_n^{(t)}$. And the final type of updates is passing messages backward in time from $x_n^{(t+1)}$ to $x_n^{(t)}$ through $f_n^{(t)}$. The details for finding the "forward" and "backward" message passing steps are also shown in Appendix A.

We schedule the messages by moving forward in time first, where we run the "forward" step starting at $t = 1$ all the way to $t = T$. We then perform the "within" step for all time frames, this step updates $\boldsymbol{r}^{(t)}$, $\boldsymbol{\tau}_r^{(t)}$, $\boldsymbol{x}^{(t)}$ and $\boldsymbol{\tau}_x^{(t)}$ that are needed for the "forward "and "backward" message passing steps. Finally we pass the messages backward in time using the "backward" step, starting at $t = T$ and ending at $t = 1$. Based on this message

$$\gamma_n^{i+1} = \frac{1}{T} \left[|\hat{x}_n^{(1)}|^2 + \tau_{x_n}^{(1)} + \frac{1}{1-\beta^2} \sum_{t=2}^{T} |\hat{x}_n^{(1)}|^2 + \tau_{x_n}^{(t)} + \beta(|\hat{x}_n^{(t-1)}|^2 + \tau_{x_n}^{(t-1)}) - 2\beta(\hat{x}_n^{(t)} \hat{x}_n^{(t-1)} + \beta \tau_{x_n}^{(t-1)}) \right]. \quad (25)$$

TABLE II
GGAMP-TSBL ALGORITHM

Definitions

$$F(\boldsymbol{r}^{k(t)},\boldsymbol{\tau}_r^{k(t)}) = \frac{\frac{\boldsymbol{r}^{k(t)}}{\boldsymbol{\tau}_r^{k(t)}}+\frac{\boldsymbol{\eta}^{k(t)}}{\boldsymbol{\psi}^{k(t)}}+\frac{\boldsymbol{\theta}^{k(t)}}{\boldsymbol{\phi}^{k(t)}}}{\frac{1}{\boldsymbol{\tau}_r^{k(t)}}+\frac{1}{\boldsymbol{\psi}^{k(t)}}+\frac{1}{\boldsymbol{\phi}^{k(t)}}} \quad \text{(D1)}$$

$$G(\boldsymbol{r}^{k(t)},\boldsymbol{\tau}_r^{k(t)}) = \frac{1}{\frac{1}{\boldsymbol{\tau}_r^{k(t)}}+\frac{1}{\boldsymbol{\psi}^{k(t)}}+\frac{1}{\boldsymbol{\phi}^{k(t)}}} \quad \text{(D2)}$$

Initialization

$\boldsymbol{S} \leftarrow |\boldsymbol{A}|^2$ (component wise magnitude squared) (N1)

Initialize $\forall t : \breve{\boldsymbol{\tau}}_x^{0(t)}, \boldsymbol{\gamma}^0 > 0$, $\breve{\boldsymbol{s}}^{0(t)} \leftarrow \boldsymbol{0}$ and $\breve{\boldsymbol{x}}^{0(t)} \leftarrow \boldsymbol{0}$ (N3)

for $i = 1, 2, ..., I_{\max}$

Initialize $\forall t : \boldsymbol{\tau}_x^{1(t)} \leftarrow \breve{\boldsymbol{\tau}}_x^{i-1(t)}, \hat{\boldsymbol{x}}^{1(t)} \leftarrow \breve{\boldsymbol{x}}^{i-1(t)}$, $\boldsymbol{s}^{1(t)} \leftarrow \breve{\boldsymbol{s}}^{i-1(t)}$

E-Step approximation

for $k = 1, 2, ..., K_{\max}$

$\boldsymbol{\eta}^{k(1)} \leftarrow 0$ (E1)

$\boldsymbol{\psi}^{k(1)} \leftarrow \boldsymbol{\gamma}^i$ (E2)

for $t = 2 : T$

$\boldsymbol{\eta}^{k(t)} \leftarrow \beta\left(\frac{\boldsymbol{r}^{k(t-1)}}{\boldsymbol{\tau}_r^{k(t-1)}}+\frac{\boldsymbol{\eta}^{k(t-1)}}{\boldsymbol{\psi}^{k(t-1)}}\right)\left(\frac{\boldsymbol{\psi}^{k(t-1)}\boldsymbol{\tau}_r^{k(t-1)}}{\boldsymbol{\psi}^{k(t-1)}+\boldsymbol{\tau}_r^{k(t-1)}}\right)$ (E3)

$\boldsymbol{\psi}^{k(t)} \leftarrow \beta^2\left(\frac{\boldsymbol{\psi}^{k(t-1)}\boldsymbol{\tau}_r^{k(t-1)}}{\boldsymbol{\psi}^{k(t-1)}+\boldsymbol{\tau}_r^{k(t-1)}}\right)+(1-\beta^2)\boldsymbol{\gamma}^i$ (E4)

end for %end of t loop

for $t = 1 : T$

$1/\boldsymbol{\tau}_p^{k(t)} \leftarrow \boldsymbol{S}\boldsymbol{\tau}_x^{k(t)}$ (E5)

$\boldsymbol{p}^{k(t)} \leftarrow \boldsymbol{s}^{k-1(t)} + \boldsymbol{\tau}_p^{k(t)}\boldsymbol{A}\hat{\boldsymbol{x}}^{k(t)}$ (E6)

$\boldsymbol{\tau}_s^{k(t)} \leftarrow \frac{\sigma^{-2}\boldsymbol{\tau}_p^{k(t)}}{\sigma^{-2}+\boldsymbol{\tau}_p^{k(t)}}$ (E7)

$\boldsymbol{s}^{k(t)} \leftarrow (1-\theta_s)\boldsymbol{s}^{k-1(t)} + \theta_s\frac{\left(\frac{\boldsymbol{p}^{k(t)}}{\boldsymbol{\tau}_p^{k(t)}}-\boldsymbol{y}^{(t)}\right)}{(\sigma^2+1/\boldsymbol{\tau}_p^{k(t)})}$ (E8)

$1/\boldsymbol{\tau}_r^{k(t)} \leftarrow \boldsymbol{S}^\top\boldsymbol{\tau}_s^{k(t)}$ (E9)

$\boldsymbol{r}^{k(t)} \leftarrow \hat{\boldsymbol{x}}^{(t)} - \boldsymbol{\tau}_r^{k(t)}\boldsymbol{A}^\top\boldsymbol{s}^{k(t)}$ (E10)

$\boldsymbol{\tau}_x^{k+1(t)} \leftarrow G(\boldsymbol{r}^{k(t)},\boldsymbol{\tau}_r^{k(t)})$ (E11)

$\hat{\boldsymbol{x}}^{k+1(t)} \leftarrow (1-\theta_x)\hat{\boldsymbol{x}}^{k(t)} + \theta_x F_n(\boldsymbol{r}^{k(t)},\boldsymbol{\tau}_r^{k(t)})$ (E12)

end for %end of t loop

for $t = T-1 : 1$

$\boldsymbol{\theta}^{k(t)} \leftarrow \frac{1}{\beta}\left(\frac{\boldsymbol{r}^{k(t+1)}}{\boldsymbol{\tau}_r^{k(t+1)}}+\frac{\boldsymbol{\theta}^{k(t+1)}}{\boldsymbol{\phi}^{k(t+1)}}\right)\left(\frac{\boldsymbol{\phi}^{k(t+1)}\boldsymbol{\tau}_r^{k(t+1)}}{\boldsymbol{\theta}^{k(t+1)}+\boldsymbol{\tau}_r^{k(t+1)}}\right)$ (E13)

$\boldsymbol{\phi}^{k(t)} \leftarrow \frac{1}{\beta^2}\left(\frac{\boldsymbol{\phi}^{k(t+1)}\boldsymbol{\tau}_r^{k(t+1)}}{\boldsymbol{\phi}^{k(t+1)}+\boldsymbol{\tau}_r^{k(t+1)}}+(1-\beta^2)\boldsymbol{\gamma}^i\right)$ (E14)

end for %end of t loop

if $\frac{1}{T}\sum_{t=1}^{T}\left(\frac{\|\hat{\boldsymbol{x}}^{k+1(t)}-\hat{\boldsymbol{x}}^{k(t)}\|^2}{\|\hat{\boldsymbol{x}}^{k+1(t)}\|^2}\right) < \epsilon_{\text{gamp}}$, break (E15)

end for %end of k loop

$\forall t, \breve{\boldsymbol{s}}^{i(t)} \leftarrow \boldsymbol{s}^{k+1(t)}, \breve{\boldsymbol{x}}^{i(t)} \leftarrow \hat{\boldsymbol{x}}^{k+1(t)}$, $\breve{\boldsymbol{\tau}}_x^{i(t)} \leftarrow \boldsymbol{\tau}_x^{k+1(t)}$

M-step

$$\gamma_n^{i+1} = \frac{1}{T}\Bigg[|\breve{x}_n^{i(1)}|^2 + \breve{\tau}_{x_n}^{i(1)} + \sum_{t=2}^{T}\frac{|\breve{x}_n^{i(t)}|^2+\breve{\tau}_{x_n}^{i(t)}}{1-\beta^2} + \frac{\beta}{1-\beta^2}\sum_{t=2}^{T}\left(|\breve{x}_n^{i(t-1)}|^2+\breve{\tau}_{x_n}^{i(t-1)}\right) - \frac{2\beta}{1-\beta^2}\sum_{t=2}^{T}\left(\breve{x}_n^{i(t)}\breve{x}_n^{i(t-1)}+\beta\breve{x}_n^{i(t-1)}\right)\Bigg] \quad \text{(U1)}$$

if $\frac{1}{T}\sum_{t=1}^{T}\left(\frac{\|\breve{\boldsymbol{x}}^{i(t)}-\breve{\boldsymbol{x}}^{i-1(t)}\|^2}{\|\breve{\boldsymbol{x}}^{i(t)}\|^2}\right) < \epsilon_{\text{em}}$, break (U2)

end for %end of i loop

schedule, the GAMP algorithm E-step computation is summarized in Table II. In Table II we use the unparenthesized superscript to indicate the iteration index, while the parenthesized superscript indicates the time frame index. Similar to Table I, $K_{\max}$ is the maximum allowed number of GAMP iterations, ϵ_{gamp} is the GAMP normalized tolerance parameter, $I_{\max}$ is the maximum allowed number of EM iterations and ϵ_{em} is the EM normalized tolerance parameter. In Table II all vector squares, divisions and multiplications are taken element wise.

The algorithm proposed can be considered an extension of the previously proposed AMP TSBL algorithm in [33]. The extension to GGAMP-TSBL includes removing the averaging of the matrix $\boldsymbol{A}$ in the derivation of the algorithm, and it includes introducing the same damping strategy used in the SMV case to improve convergence. The complexity of the GGAMP-TSBL algorithm is also dominated by the E-step which in turn is dominated by matrix multiplications by $\boldsymbol{A}$, $\boldsymbol{A}^\top$, $\boldsymbol{S}$ and $\boldsymbol{S}^\top$, implying that the computational cost is $\mathcal{O}(MN)$ flops per iteration per frame. Therefore the complexity of the proposed algorithm is $\mathcal{O}(TMN)$ multiplied by the total number of GAMP algorithm iterations.

C. GGAMP-TSBL M-Step

Upon the convergence of the E-step, the M-step learns $\boldsymbol{\gamma}$ from the data by treating $\boldsymbol{x}$ as a hidden variable and then maximizing $\mathbb{E}_{\bar{\boldsymbol{x}}|\bar{\boldsymbol{y}};\boldsymbol{\gamma}^i,\sigma^2,\beta}[\log p(\bar{\boldsymbol{y}},\bar{\boldsymbol{x}},\boldsymbol{\gamma};\sigma^2,\beta)]$.

$$\boldsymbol{\gamma}^{i+1} = \underset{\boldsymbol{\gamma}}{\operatorname{argmin}}\,\mathbb{E}_{\bar{\boldsymbol{x}}|\bar{\boldsymbol{y}};\boldsymbol{\gamma}^i,\sigma^2,\beta}[-\log p(\bar{\boldsymbol{y}},\bar{\boldsymbol{x}},\boldsymbol{\gamma};\sigma^2,\beta)].$$

The derivation of γ_n^{i+1} M-step update follows the same steps as the SMV case. The derivation is omitted here due to space limitation, and γ_n^{i+1} update is given in (25) at the bottom of the previous page. We note here that the M-step $\boldsymbol{\gamma}$ learning rule in (25) is the same as the one derived in [36]. Both algorithms use the same AR(1) model for $\boldsymbol{x}^{(t)}$, but they differ in the implementation of the E-step. In the case that the correlation coefficient β or the noise variance σ^2 are unknown, the EM algorithm can be used to estimate their values as well.

V. NUMERICAL RESULTS

In this section we present a numerical study to illustrate the performance and complexity of the proposed GGAMP-SBL and GGAMP-TSBL algorithms. The performance and complexity were studied through two metrics. The first metric studies the ability of the algorithm to recover $\boldsymbol{x}$, for which we use the normalized mean squared error NMSE in the SMV case:

$$\text{NMSE} \triangleq \|\hat{\boldsymbol{x}}-\boldsymbol{x}\|^2/\|\boldsymbol{x}\|^2,$$

and the time-averaged normalized mean squared error TNMSE in the MMV case:

$$\text{TNMSE} \triangleq \frac{1}{T}\sum_{t=1}^{T}\|\hat{\boldsymbol{x}}^{(t)}-\boldsymbol{x}^{(t)}\|^2/\|\boldsymbol{x}^{(t)}\|^2.$$

The second metric studies the complexity of the algorithm by tracking the time the algorithm requires to compute the final estimate $\hat{\boldsymbol{x}}$. We measure the time in seconds. While the absolute runtime could vary if the same experiments were to be run on a different machine, the runtimes of the algorithms of interest in relationship to each other is a good estimate of the relative computational complexity.

Several types of non-i.i.d.-Gaussian matrix were used to explore the robustness of the proposed algorithms relative to the standard SBL and TMSBL. The four different types of matrices are similar to the ones previously used in [26] and are described as follows:

-*Column correlated matrices:* The rows of $\boldsymbol{A}$ are independent zero-mean Gaussian Markov processes with the following correlation coefficient $\rho = \mathbb{E}\{\boldsymbol{a}_{.,n}^\top\boldsymbol{a}_{.,n+1}\}/\mathbb{E}\{|\boldsymbol{a}_{.,n}|^2\}$, where

$\boldsymbol{a}_{.,n}$ is the nth column of $\boldsymbol{A}$. In the experiments the correlation coefficient ρ is used as the measure of deviation from the i.i.d.-Gaussian matrix.

-Low rank product matrices: We construct a rank deficient $\boldsymbol{A}$ by $\boldsymbol{A} = \frac{1}{N}\boldsymbol{H}\boldsymbol{G}$ with $\boldsymbol{H} \in \mathbb{R}^{M\times R}$, $\boldsymbol{G} \in \mathbb{R}^{R\times N}$ and $R < M$. The entries of $\boldsymbol{H}$ and $\boldsymbol{G}$ are i.i.d.-Gaussian with zero mean and unit variance. The rank ratio R/N is used as the measure of deviation from the i.i.d.-Gaussian matrix.

-Ill conditioned matrices: we construct $\boldsymbol{A}$ with a condition number $\kappa > 1$ as follows. $\boldsymbol{A} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^\top$, where $\boldsymbol{U}$ and $\boldsymbol{V}^\top$ are the left and right singular vector matrices of an i.i.d.-Gaussian matrix, and $\boldsymbol{\Sigma}$ is a singular value matrix with $\Sigma_{i,i}/\Sigma_{i+1,i+1} = \kappa^{1/(M-1)}$ for $i = 1, 2, ..., M-1$. The condition number κ is used as the measure of deviation from the i.i.d.-Gaussian matrix.

-Non-zero mean matrices: The elements of $\boldsymbol{A}$ are $a_{m,n} \sim \mathcal{N}(\mu, \frac{1}{N})$. The mean μ is used as a measure of deviation from the zero-mean i.i.d.-Gaussian matrix. It is worth noting that in the case of non-zero mean $\boldsymbol{A}$, convergence of the GGAMP-SBL is not enhanced by damping but more by the mean removal procedure explained in [26]. We include it in the implementation of our algorithm, and we include it in the numerical results to make the study more inclusive of different types of generic $\boldsymbol{A}$ matrices.

Although we have provided an estimation procedure, based on the EM algorithm, for the noise variance σ^2 in (13), in all experiments we assume that the noise variance σ^2 is known. We also found that the SBL algorithm does not necessarily have the best performance when the exact σ^2 is used, and in our case, it was empirically found that using an estimate $\hat{\sigma}^2 = 3\sigma^2$ yields better results. Therefore $\hat{\sigma}^2$ is used for SBL, TMSBL, GGAMP-SBL and GGAMP-TSBL throughout our experiments.

A. SMV GGAMP-SBL Numerical Results

In this section we compare the proposed SMV algorithm (GGAMP-SBL) against the original SBL and against two AMP algorithms that have shown improvement in robustness over the original AMP/GAMP, namely the SwAMP algorithm [27] and the MADGAMP algorithm [26]. As a performance benchmark, we use a lower bound on the achievable NMSE which is similar to the one in [26]. The bound is found using a "genie" that knows the support of the sparse vector $\boldsymbol{x}$. Based on the known support, $\bar{\boldsymbol{A}}$ is constructed from the columns of $\boldsymbol{A}$ corresponding to non-zero elements of $\boldsymbol{x}$, and an MMSE solution using $\bar{\boldsymbol{A}}$ is computed.

$$\hat{\boldsymbol{x}} = \bar{\boldsymbol{A}}^\top(\bar{\boldsymbol{A}}\bar{\boldsymbol{A}}^\top + \sigma^2\boldsymbol{I})^{-1}\boldsymbol{y}.$$

In all SMV experiments, $\boldsymbol{x}$ had exactly K non-zero elements in random locations, and the nonzero entires were drawn independently from a zero-mean unit-variance Gaussian distribution. In accordance with the model (1), an AWGN channel was used with the SNR defined by:

$$\text{SNR} \triangleq \mathbb{E}\{\|\boldsymbol{A}\boldsymbol{x}\|^2\}/\mathbb{E}\{\|\boldsymbol{y} - \boldsymbol{A}\boldsymbol{x}\|^2\}.$$

1) Robustness to Generic Matrices at High SNR: The first experiment investigates the robustness of the proposed algorithm to generic $\boldsymbol{A}$ matrices. It compares the algorithms of interest using the four types of matrices mentioned above, over a range of deviation from the i.i.d.-Gaussian case. For each matrix type, we start with an i.i.d.-Gaussian $\boldsymbol{A}$ and increase the deviation over 11 steps. We monitor how much deviation the different algorithms can tolerate, before we start seeing significant performance degradation compared to the "genie" bound. The vector $\boldsymbol{x}$ was drawn from a Bernoulli-Gaussian distribution with non-zero probability $\lambda = 0.2$, with $N = 1000$, $M = 500$ and SNR $= 60$ dB.

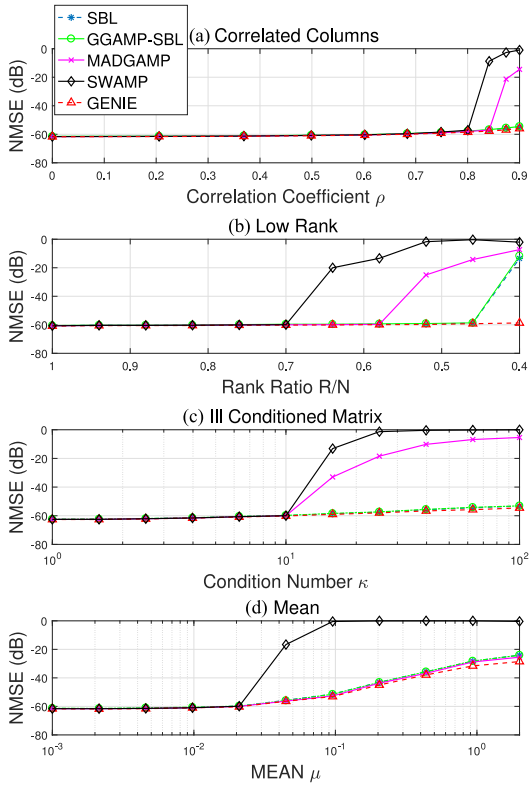


Fig. 5. NMSE comparison of SMV algorithms under non-i.i.d.-Gaussian $\boldsymbol{A}$ matrices with SNR $= 60$ dB.

The NMSE results in Fig. 5 show that the performance of GGAMP-SBL was able to match that of the original SBL even for $\boldsymbol{A}$ matrices with the most deviation from the i.i.d.-Gaussian case. Both algorithms nearly achieved the bound in most cases, with the exception when the matrix is low rank with a rank ratio less than 0.45 where both algorithms fail to achieve the bound. This supports the evidence we provided before for the convergence of the GGAMP-SBL algorithm, which predicted its ability to match the performance of the original SBL. As for other AMP implementations, despite the improvement in robustness they provide over traditional AMP/GAMP, they cannot guarantee convergence beyond a certain point, and their robustness is surpassed by GGAMP-SBL in most cases. The only exception is when $\boldsymbol{A}$ is non-zero mean, where the GGAMP-SBL and the MADGAMP algorithms share similar performance. This is due to the fact that both algorithms use mean removal to transform the problem into a zero-mean equivalent problem, which both algorithms can handle well.

The complexity of the GGAMP-SBL algorithm is studied in Fig. 6. The figure shows how the GGAMP-SBL was able to reduce the complexity compared to the original SBL implementation. It also shows that even when the algorithm is slowed

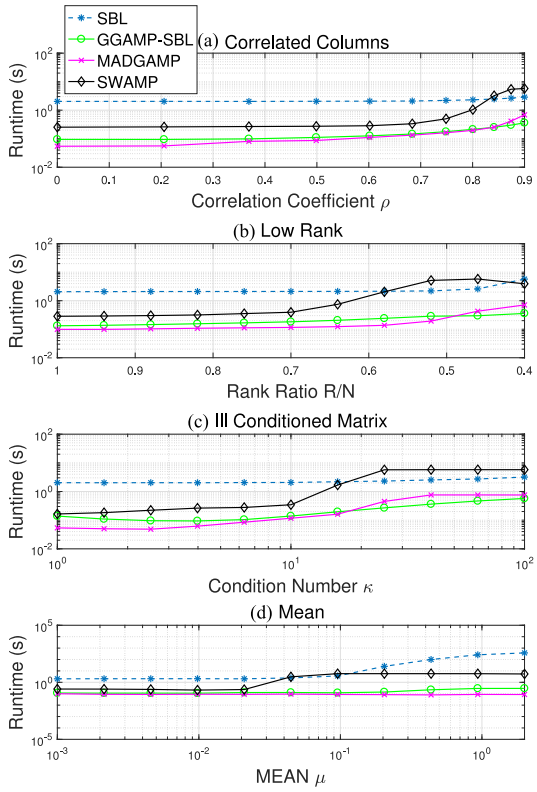


Fig. 6. Runtime comparison of SMV algorithms under non-i.i.d.-Gaussian $\boldsymbol{A}$ matrices with SNR $= 60$ dB.

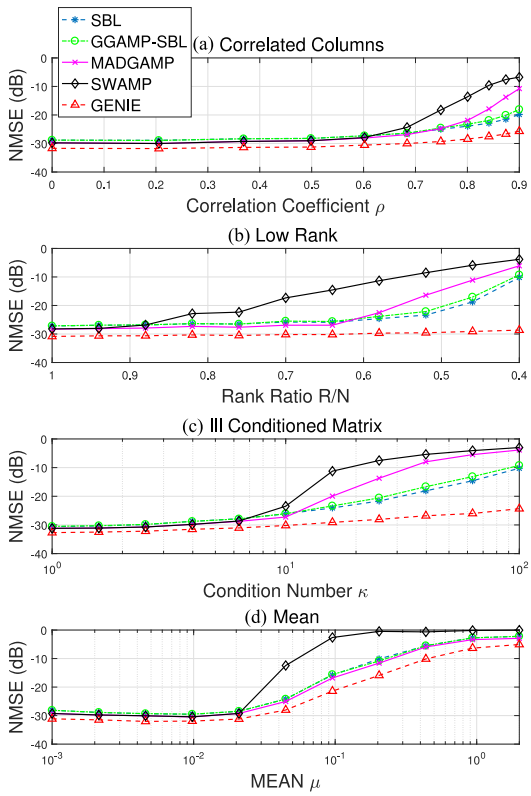


Fig. 7. NMSE comparison of SMV algorithms under non-i.i.d.-Gaussian $\boldsymbol{A}$ matrices with SNR $= 30$ dB.

down by heavier damping, the algorithm still has faster runtimes than the original SBL.

2) Robustness to Generic Matrices at Lower SNR: In this experiment we examine the performance and complexity of the proposed algorithm at a lower SNR setting than the previous experiment. We lower the SNR to 30 dB and collect the same data points as in the previous experiment. The results in Fig. 7 show that the performance of the GGAMP-SBL algorithm is still generally matching that of the original SBL algorithm with slight degradation. The MADGAMP algorithm provides slightly better performance than both SBL algorithms when the deviation from the i.i.d.-sub-Gaussian case is not too large. This can be due to the fact that we choose to run the MADGAMP algorithm with exact knowledge of the data model rather than learn the model parameters, while both SBL algorithms have information about the noise variance only. As the deviation in $\boldsymbol{A}$ increases, GGAMP-SBL's performance surpasses MADGAMP and SWAMP algorithms, providing better robustness at lower SNR.

On the complexity side, we see from Fig. 8 that the GGAMP-SBL continues to have reduced complexity compared to the original SBL.

3) Performance and Complexity Versus Problem Dimensions: To show the effect of increasing the problem dimensions on the performance and complexity of the different algorithms, we plot the NMSE and runtime against N, while we keep an M/N ratio of 0.5, a K/N ratio of 0.2 and an SNR of 60 dB. We run the experiment using column correlated matrices with $\rho = 0.9$.

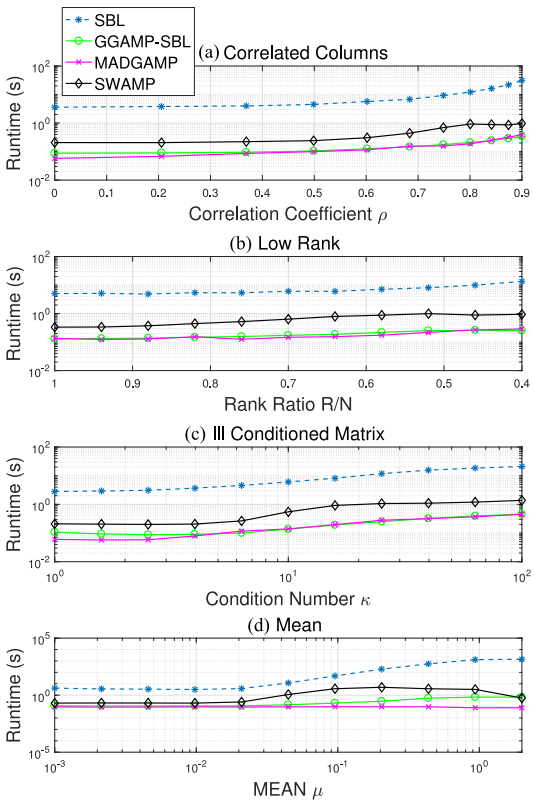
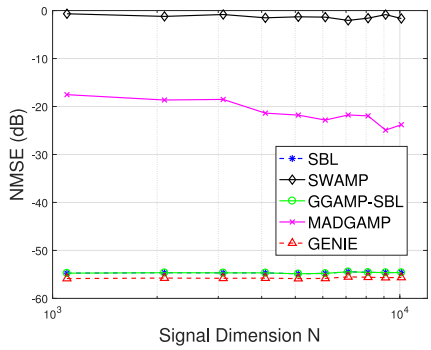
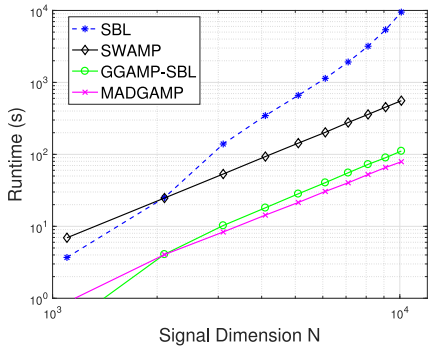


Fig. 8. Runtime comparison of SMV algorithms under non-i.i.d.-Gaussian $\boldsymbol{A}$ matrices with SNR $= 30$ dB.



(a) NMSE versus N



(b) Runtime versus N

Fig. 9. Performance and complexity comparison for SMV algorithms versus problem dimensions.

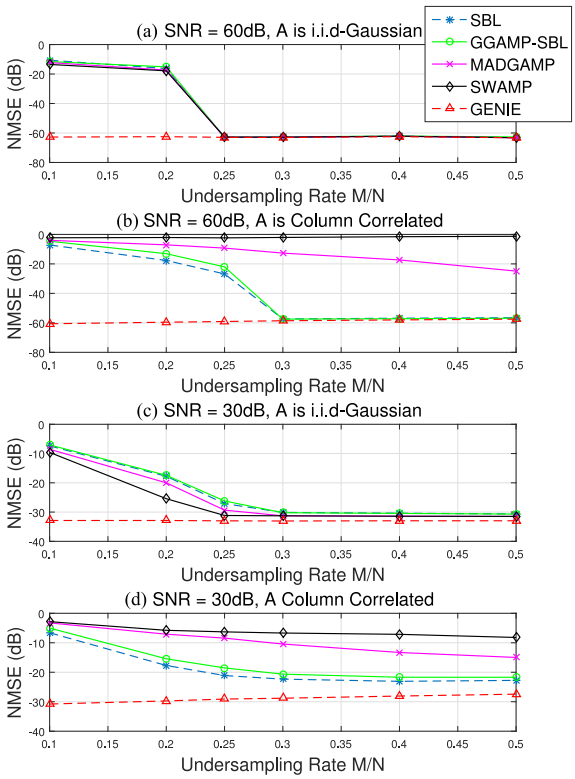


Fig. 10. NMSE comparison of SMV algorithms versus the undersampling rate M/N.

As expected from previous experiments, Fig. 9(a) shows that only GGAMP-SBL and SBL algorithms can recover $\boldsymbol{x}$ when we use column correlated matrices with a correlation coefficient of $\rho = 0.9$. The comparison between the performance of SBL and GGAMP-SBL show almost identical NMSE.

As problem dimensions grow, Fig. 9(b) shows that the difference in runtimes between the original SBL and GGAMP-SBL algorithms grows to become more significant, which suggests that the GGAMP-SBL is more practical for large size problems.

4) Performance Versus Undersampling Ratio M/N: In this section we examine the ability of the proposed algorithm to recover a sparse vector from undersampled measurements at different undersampling ratios M/N. In the below experiments we fix N at 1000 and vary M. We set the Bernoulli-Gaussian non-zero probability λ so that M/K has an average of three measurements for each non-zero component. We plot the NMSE versus the undersampling ratio M/N for i.i.d.-Gaussian matrices $\boldsymbol{A}$ and for column correlated $\boldsymbol{A}$ with $\rho = 0.9$. We run the experiments at SNR $= 60$ dB and at SNR $= 30$ dB. In Fig. 10 we find that for SNR $= 60$ dB and i.i.d.-Gaussian $\boldsymbol{A}$, all algorithms meet the SKS bound when the undersampling ratio is larger than or equal to 0.25, while all algorithms fail to meet the bound at any ratio smaller than that. When $\boldsymbol{A}$ is column correlated, SBL and GGAMP-SBL are able to meet the SKS bound at $M/N \geq 0.3$, while MADGAMP and SwAMP do not meet the bound even at $M/N = 0.5$. We also note the MADGAMP's NMSE slowly improves with increased underasampling ratio, while SwAMP's NMSE does not. At SNR $= 30$ dB, with i.i.d.-Gaussian $\boldsymbol{A}$ all algorithms are close to the SKS bound when the undersampling ratio is larger than 0.3. At $M/N \leq 0.3$, SBL and GGAMP-SBL are slightly outperformed by MADGAMP, while SwAMP seems to have the best performance in this region. When $\boldsymbol{A}$ is column correlated, NMSE of SBL and GGAMP-SBL outperform the other two algorithms, and similar to the SNR $= 60$ dB case, MADGAMP's NMSE seems to slowly improve with increased undersampling ratio, while SwAMP's NMSE does not improve.

B. MMV GGAMP-TSBL Numerical Results

In this section, we present a numerical study to illustrate the performance and complexity of the proposed GGAMP-TSBL algorithm. Although the AMP MMV algorithm in [37] can be extended to incorporate damping, the current implementation of AMP MMV does not include damping and will diverge when used with the type of generic $\boldsymbol{A}$ matrices we are considering for our experiments. Therefore, we restrict the comparison of the performance and complexity of the GGAMP-TSBL algorithm to the TMSBL algorithm. We also compare the recovery performance against a lower bound on the achievable TNMSE by extending the support aware Kalman smoother (SKS) from [37] to include damping and hence be able to handle generic $\boldsymbol{A}$ matrices. The implementation of the smoother is straight forward, and is exactly the same as the E-step part in Table II, when the true values of σ^2, $\boldsymbol{\gamma}$ and β are used, and when $\boldsymbol{A}$ is modified to include only the columns corresponding to the non-zero elements in $\boldsymbol{x}^{(t)}$. An AWGN channel was also assumed in the case of MMV.

1) Robustness to Generic Matrices at High SNR: The experiment investigates the robustness of the proposed algorithm

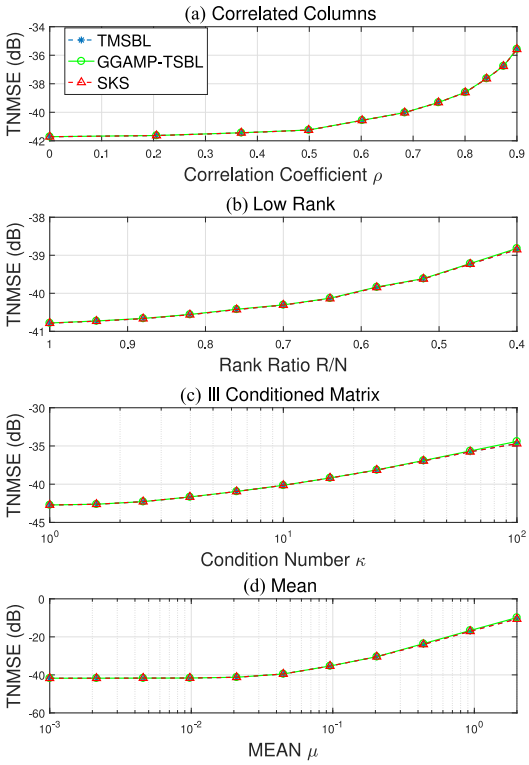


Fig. 11. TNMSE comparison of MMV algorithms under non-i.i.d.-Gaussian $\boldsymbol{A}$ matrices with SNR = 60 dB.

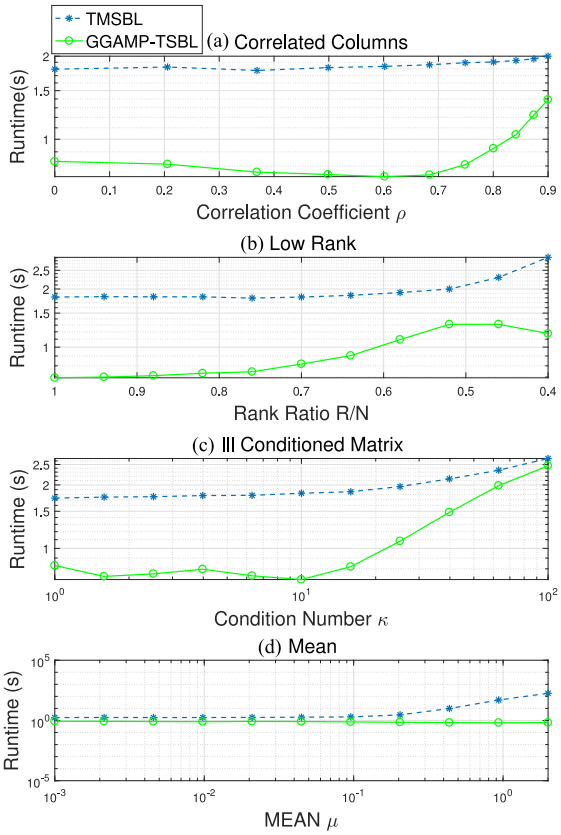


Fig. 12. Runtime comparison of MMV algorithms under non-i.i.d.-Gaussian $\boldsymbol{A}$ matrices with SNR = 60 dB.

by comparing it to the TMSBL and the support aware smoother. Once again we use the four types of matrices mentioned at the beginning of this section, over the same range of deviation from the i.i.d.-Gaussian case. For this experiment we set $N = 1000$, $M = 500$, $\lambda = 0.2$, $SNR = 60$ dB and the temporal correlation coefficient β to 0.9. We choose a relatively high value for β to provide large deviation from the SMV case. This is due to the fact that the no correlation case is reduced to solving multiple SMV instances in the E-step, and then applying the M-step to update the hyperparameter vector $\boldsymbol{\gamma}$, which is common across time frames [35]. The TNMSE results in Fig. 11 show that the performance of GGAMP-TSBL was able to match that of TMSBL in all cases and they both achieved the SKS bound.

Once again Fig. 12 shows that the proposed GGAMP-TSBL was able to reduce the complexity compared to the TMSBL algorithm, even when damping was used. Although the complexity reduction does not seem to be significant for the selected problem size and SNR, we will see in the following experiments how this reduction becomes more significant as the problem size grows or as a lower SNR is used.

2) Robustness to Generic Matrices at Lower SNR: The performance and complexity of the proposed algorithm are examined at a lower SNR setting than the previous experiment. We set the SNR to 30 dB and collect the same data points collected as in the 60 dB SNR case. Fig. 13 shows that the GGAMP-TSBL performance matches that of the TMSBL and almost achieves the bound in most cases.

Similar to the previous cases, Fig. 14 shows that the complexity of GGAMP-TSBL is lower than that of TMSBL.

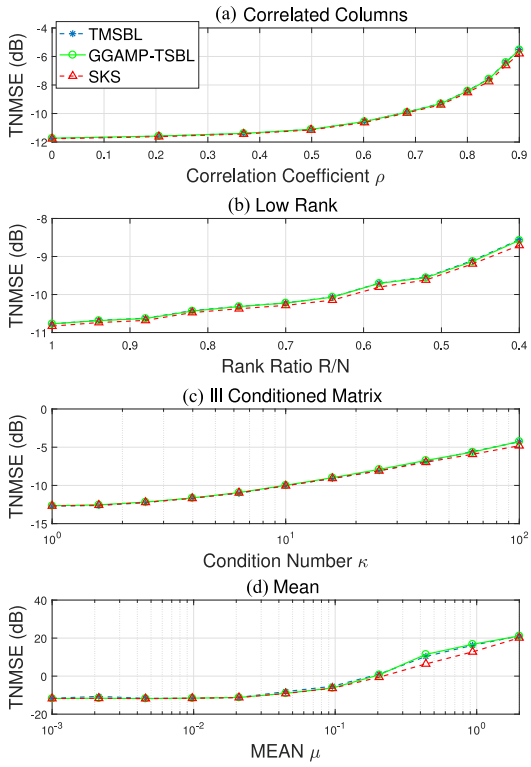


Fig. 13. TNSME comparison of MMV algorithms under non-i.i.d.-Gaussian $\boldsymbol{A}$ matrices with SNR = 30 dB.

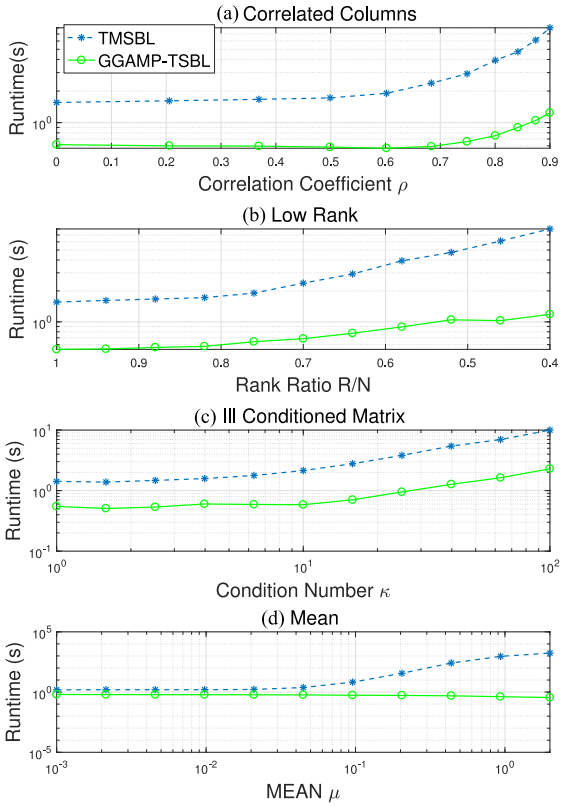


Fig. 14. Runtime comparison of MMV algorithms under non-i.i.d.-Gaussian $\boldsymbol{A}$ matrices with SNR $= 30$ dB.

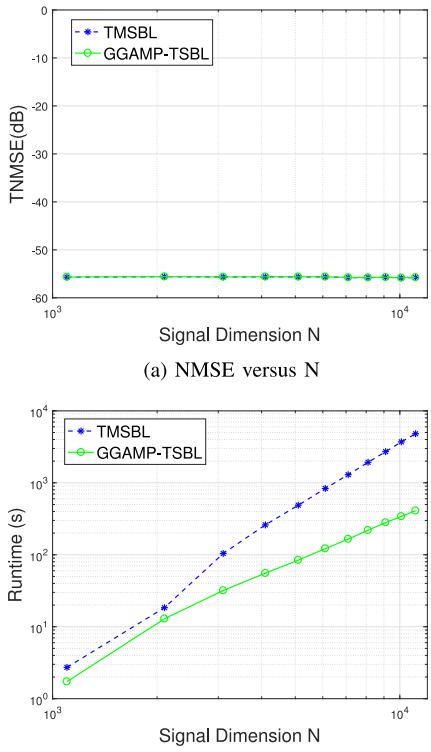


Fig. 15. Performance and complexity comparison for MMV algorithms versus problem dimensions.

3) Performance and Complexity Versus Problem Dimension: To validate the claim that the proposed algorithm is more suited to deal with large scale problems we study the algorithms' performance and complexity against the signal dimension N. We keep an M/N ratio of 0.5, a K/N ratio of 0.2 and an SNR of 60 dB. We run the experiment using column correlated matrices with $\rho = 0.9$. In addition, we set β to 0.9, high temporal correlation. In terms of performance, Fig. 15(a) shows that the proposed GGAMP-TSBL algorithm was able to match the performance of TMSBL. However, in terms of complexity, similar to the SMV case, Fig. 15(b) shows that the runtime difference becomes more significant as the problem size grows, making the GGAMP-SBL a better choice for large scale problems.

VI. Conclusion

In this paper, we presented a GAMP based SBL algorithm for solving the sparse signal recovery problem. SBL uses sparsity promoting priors on $\boldsymbol{x}$ that admit a Gaussian scale mixture representation. Because of the Gaussian embedding offered by the GSM class of priors, we were able to leverage the Gaussian GAMP algorithm along with it's convergence guarantees given in [24], when sufficient damping is used, to develop a reliable and fast algorithm. We numerically showed how this damped GGAMP implementation of the SBL algorithm also reduces the cost function of the original SBL approach. The algorithm was then extended to solve the MMV SSR problem in the case of generic $\boldsymbol{A}$ matrices and temporal correlation, using a similar GAMP based SBL approach. Numerical results show that both the SMV and MMV proposed algorithms were more robust to generic $\boldsymbol{A}$ matrices when compared to other AMP algorithms. In addition, numerical results also show the significant reduction in complexity the proposed algorithms offer over the original SBL and TMSBL algorithms, even when sufficient damping is used to slow down the updates to guarantee convergence. Therefore the proposed algorithms address the convergence limitations in AMP algorithms as well as the complexity challenges in traditional SBL algorithms, while retaining the positive attributes namely the robustness of SBL to generic $\boldsymbol{A}$ matrices, and the low complexity of message passing algorithms.

Appendix A
Derivation of GGAMP-TSBL Updates

A. The Within Step Updates

To make the factor graph for the within step in Fig. 4 exactly the same as the SMV factor graph we combine the product of the two messages incoming from $f_n^{(t)}$ and $f_n^{(t+1)}$ to $x_n^{(t)}$ into one message as follows:

$$
\begin{aligned}
V_{f_n^{(t)}\to x_n^{(t)}} &\propto \mathcal{N}(x_n^{(t)};\eta_n^{(t)},\psi_n^{(t)}) \\
V_{f_n^{(t+1)}\to x_n^{(t)}} &\propto \mathcal{N}(x_n^{(t)};\theta_n^{(t)},\phi_n^{(t)}) \\
V_{\bar{f}_n^{(t)}\to x_n^{(t)}} &\propto \mathcal{N}(x_n^{(t)};\rho_n^{(t)},\zeta_n^{(t)}) \\
&\propto \mathcal{N}\left(x_n^{(t)};\frac{\frac{\eta_n^{(t)}}{\psi_n^{(t)}}+\frac{\theta_n^{(t)}}{\phi_n^{(t)}}}{\frac{1}{\psi_n^{(t)}}+\frac{1}{\phi_n^{(t)}}},\frac{1}{\frac{1}{\psi_n^{(t)}}+\frac{1}{\phi_n^{(t)}}}\right). \qquad (26)
\end{aligned}
$$

Combining these two messages reduces each time frame factor graph to an equivalent one to the SMV case with a modified prior on $x_n^{(t)}$ of (26). Applying the damped GAMP algorithm from [24] with $p(x_n^{(t)})$ given in (26):

$$g_x^{(t)} = \frac{\frac{\boldsymbol{r}^{(t)}}{\boldsymbol{\tau}_r^{(t)}} + \frac{\boldsymbol{\rho}^{(t)}}{\boldsymbol{\zeta}^{(t)}}}{\frac{1}{\boldsymbol{\tau}_r^{(t)}} + \frac{1}{\boldsymbol{\zeta}^{(t)}}} = \frac{\frac{\boldsymbol{r}^{(t)}}{\boldsymbol{\tau}_r^{(t)}} + \frac{\boldsymbol{\eta}^{(t)}}{\boldsymbol{\psi}^{(t)}} + \frac{\boldsymbol{\theta}^{(t)}}{\boldsymbol{\phi}^{(t)}}}{\frac{1}{\boldsymbol{\tau}_r^{(t)}} + \frac{1}{\boldsymbol{\psi}^{(t)}} + \frac{1}{\boldsymbol{\phi}^{(t)}}}$$

$$\boldsymbol{\tau}_x^{(t)} = \frac{1}{\frac{1}{\boldsymbol{\tau}_r^{(t)}} + \frac{1}{\boldsymbol{\zeta}^{(t)}}} = \frac{1}{\frac{1}{\boldsymbol{\tau}_r^{(t)}} + \frac{1}{\boldsymbol{\psi}^{(t)}} + \frac{1}{\boldsymbol{\phi}^{(t)}}}.$$

B. Forward Message Updates

$$\begin{aligned}
V_{f_n^{(1)}\to x_n^{(1)}} &\propto \mathcal{N}(x_n^{(1)}; 0, \gamma_n) \\
V_{f_n^{(t)}\to x_n^{(t)}} &\propto \mathcal{N}(x_n^{(t)}; \eta_n^{(t)}, \psi_n^{(t)}) \\
&\propto \int \left(\prod_{l=1}^{M} V_{g_l^{(t-1)}\to x_n^{(t-1)}}\right) V_{f_n^{(t-1)}\to x_n^{(t-1)}} \\
&\qquad P(x_n^{(t)} \mid x_n^{(t-1)})\, \mathrm{d}x_n^{(t-1)} \\
&\propto \int \mathcal{N}(x_n^{(t-1)}; r_n^{(t-1)}, \tau_{r_n}^{(t-1)}) \\
&\qquad \mathcal{N}(x_n^{(t-1)}; \eta_n^{(t-1)}, \psi_n^{(t-1)}) \\
&\qquad \mathcal{N}(x_n^{(t)}; \beta x_n^{(t-1)}, (1-\beta^2)\gamma_n)\, \mathrm{d}x_n^{(t-1)}.
\end{aligned}$$

Using rules for Gaussian pdf multiplication and convolution we get the $\eta_n^{(t)}$ and $\psi_n^{(t)}$ updates given in Table II (E3) and (E4).

C. Backward Message Updates

$$\begin{aligned}
V_{f_n^{(t+1)}\to x_n^{(t)}} &\propto \mathcal{N}(x_n^{(t)}; \theta_n^{(t)}, \phi_n^{(t)}) \\
&\propto \int \left(\prod_{l=1}^{M} V_{g_l^{(t+1)}\to x_n^{(t+1)}}\right) V_{f_n^{(t+2)}\to x_n^{(t+1)}} \\
&\qquad P(x_n^{(t+1)} \mid x_n^{(t)})\, \mathrm{d}x_n^{(t+1)} \\
&\propto \int \mathcal{N}(x_n^{(t+1)}; r_n^{(t+1)}, \tau_{r_n}^{(t+1)}) \\
&\qquad \mathcal{N}(x_n^{(t+1)}; \theta_n^{(t+1)}, \phi_n^{(t+1)}) \\
&\qquad \mathcal{N}(x_n^{(t+1)}; \beta x_n^{(t)}, (1-\beta^2)\gamma_n)\, \mathrm{d}x_n^{(t+1)}.
\end{aligned}$$

Using rules for Gaussian pdf multiplication and convolution we get the $\theta_n^{(t)}$ and $\phi_n^{(t)}$ updates given in Table II (E13) and (E14).

REFERENCES

[1] D. L. Donoho, "Compressed sensing," *IEEE Trans. Inf. Theory*, vol. 52, no. 4, pp. 1289–1306, Apr. 2006.

[2] E. J. Candes and T. Tao, "Near-optimal signal recovery from random projections: Universal encoding strategies?," *IEEE Trans. Inf. Theory*, vol. 52, no. 12, pp. 5406–5425, Dec. 2006.

[3] R. G. Baraniuk, "Compressive sensing," *IEEE Signal Process. Mag.*, vol. 24, no. 4, pp. 118–121, Jul. 2007.

[4] M. Elad, *Sparse and Redundant Representations: From Theory to Applications in Signal and Image Processing*. New York, NY, USA: Springer, 2010.

[5] S. Foucart and H. Rauhut, *A Mathematical Introduction to Compressive Sensing (Applied and Numerical Harmonic Analysis)*. Cambridge, MA, USA: Birkhäuser, 2013.

[6] Y. C. Eldar and G. Kutyniok, eds., *Compressed Sensing: Theory and Applications*. Cambridge, U.K.: Cambridge Univ. Press, 2012.

[7] B. K. Natarajan, "Sparse approximate solutions to linear systems," *SIAM J. Comput.*, vol. 24, no. 2, pp. 227–234, 1995.

[8] E. J. Candes and T. Tao, "Decoding by linear programming," *IEEE Trans. Inf. Theory*, vol. 51, no. 12, pp. 4203–4215, Dec. 2005.

[9] S. S. Chen, D. L. Donoho, and M. A. Saunders, "Atomic decomposition by basis pursuit," *SIAM Rev.*, vol. 43, no. 1, pp. 129–159, 2001.

[10] J. A. Tropp and A. C. Gilbert, "Signal recovery from random measurements via orthogonal matching pursuit," *IEEE Trans. Inf. Theory*, vol. 53, no. 12, pp. 4655–4666, Dec. 2007.

[11] M. A. Figueiredo, J. M. Bioucas-Dias, and R. D. Nowak, "Majorization-minimization algorithms for wavelet-based image restoration," *IEEE Trans. Image Process.*, vol. 16, no. 12, pp. 2980–2991, Dec. 2007.

[12] E. J. Candes, M. B. Wakin, and S. P. Boyd, "Enhancing sparsity by reweighted ℓ_1 minimization," *J. Fourier Anal. Appl.*, vol. 14, no. 5–6, pp. 877–905, 2008.

[13] R. Chartrand and W. Yin, "Iteratively reweighted algorithms for compressive sensing," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, 2008, pp. 3869–3872.

[14] M. E. Tipping, "Sparse Bayesian learning and the relevance vector machine," *J. Mach. Learn. Res.*, vol. 1, pp. 211–244, Sept. 2001.

[15] D. P. Wipf and B. D. Rao, "Sparse Bayesian learning for basis selection," *IEEE Trans. Signal Process.*, vol. 52, pp. 2153–2164, Aug. 2004.

[16] J. P. Vila and P. Schniter, "Expectation-maximization Gaussian-mixture approximate message passing," *IEEE Trans. Signal Process.*, vol. 61, no. 19, pp. 4658–4672, Oct. 2013.

[17] S. D. Babacan, R. Molina, and A. K. Katsaggelos, "Bayesian compressive sensing using laplace priors," *IEEE Trans. Image Process.*, vol. 19, no. 1, pp. 53–63, Jan. 2010.

[18] L. He and L. Carin, "Exploiting structure in wavelet-based Bayesian compressive sensing," *IEEE Trans. Signal Process.*, vol. 57, no. 9, pp. 3488–3497, Sep. 2009.

[19] S. Ji, Y. Xue, and L. Carin, "Bayesian compressive sensing," *IEEE Trans. Signal Process.*, vol. 56, no. 6, pp. 2346–2356, Jun. 2008.

[20] D. Baron, S. Sarvotham, and R. G. Baraniuk, "Bayesian compressive sensing via belief propagation," *IEEE Trans. Signal Process.*, vol. 58, no. 1, pp. 269–280, Jan. 2010.

[21] D. L. Donoho, A. Maleki, and A. Montanari, "Message passing algorithms for compressed sensing: I. motivation and construction," in *Proc. IEEE Inf. Theory Workshop Inf. Theory*, Jan 2010, pp. 1–5.

[22] S. Rangan, "Generalized approximate message passing for estimation with random linear mixing," in *Proc. IEEE Int. Symp. Inf. Theory*, Jul. 2011, pp. 2168–2172.

[23] M. Bayati and A. Montanari, "The dynamics of message passing on dense graphs, with applications to compressed sensing," *IEEE Trans. Inf. Theory*, vol. 57, no. 2, pp. 764–785, Feb. 2011.

[24] S. Rangan, P. Schniter, and A. Fletcher, "On the convergence of approximate message passing with arbitrary matrices," in *Proc. IEEE Int. Symp. Inf. Theory*, Jun. 2014, pp. 236–240.

[25] F. Caltagirone, L. Zdeborova, and F. Krzakala, "On convergence of approximate message passing," in *Proc. IEEE Int. Symp. Inf. Theory*, Jun. 2014, pp. 1812–1816.

[26] J. Vila, P. Schniter, S. Rangan, F. Krzakala, and L. Zdeborov, "Adaptive damping and mean removal for the generalized approximate message passing algorithm," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process.*, Apr. 2015, pp. 2021–2025.

[27] A. Manoel, F. Krzakala, E. Tramel, and L. Zdeborovà, "Swept approximate message passing for sparse estimation," in *Proc. 32nd Int. Conf. Mach. Learn.*, 2015, pp. 1123–1132.

[28] S. Rangan, A. K. Fletcher, P. Schniter, and U. S. Kamilov, "Inference for generalized linear models via alternating directions and bethe free energy minimization," in *Proc. IEEE Int. Symp. Inf. Theory*, 2015, pp. 1640–1644.

[29] D. F. Andrews and C. L. Mallows, "Scale mixtures of normal distributions," *J. Roy. Stat. Soc. Series B (Methodological)*, vol. 36, pp. 99–102, 1974.

[30] J. Palmer, K. Kreutz-Delgado, B. D. Rao, and D. P. Wipf, "Variational EM algorithms for non-Gaussian latent variable models," in *Proc. Adv. Neural Inf. Process. Syst.*, 2005, pp. 1059–1066.

[31] K. Lange and J. S. Sinsheimer, "Normal/independent distributions and their applications in robust regression," *J. Comput. Graph. Stat.*, vol. 2, no. 2, pp. 175–198, 1993.

[32] N. L. Pedersen, C. N. Manchón, M.-A. Badiu, D. Shutin, and B. H. Fleury, "Sparse estimation using Bayesian hierarchical prior modeling for real and complex linear models," *Signal Process.*, vol. 115, pp. 94–109, 2015.
[33] M. Al-Shoukairi and B. Rao, "Sparse Bayesian learning using approximate message passing," in *Proc. IEEE 48th Asilomar Conf. Signals Syst. Comput.*, 2014, pp. 1957–1961.
[34] Z. Zhang and B. D. Rao, "Sparse signal recovery with temporally correlated source vectors using sparse Bayesian learning," *IEEE J. Sel. Top. Signal Process.*, vol. 5, no. 5, pp. 912–926, Sep. 2011.
[35] D. P. Wipf and B. D. Rao, "An empirical Bayesian strategy for solving the simultaneous sparse approximation problem," *IEEE Trans. Signal Process.*, vol. 55, no. 7, pp. 3704–3716, Jul. 2007.
[36] R. Prasad, C. R. Murthy, and B. D. Rao, "Joint approximately sparse channel estimation and data detection in OFDM systems using sparse Bayesian learning," *IEEE Trans. Signal Process.*, vol. 62, no. 14, pp. 3591–3603, Jul. 2014.
[37] J. Ziniel and P. Schniter, "Efficient high-dimensional inference in the multiple measurement vector problem," *IEEE Trans. Signal Process.*, vol. 61, no. 2, pp. 340–354, Jan. 2013.
[38] R. Giri and B. D. Rao, "Type I and type II Bayesian methods for sparse signal recovery using scale mixtures," *IEEE Trans. Signal Process.*, vol. 64, no. 13, pp. 3418–3428, 2016.
[39] M. E. Tipping *et al.*, "Fast marginal likelihood maximisation for sparse Bayesian models," in *Proc. 9th Int. Workshop Artif. Intell. Stat.*, 2003. [Online]. Available: http://dblp.org/db/conf/aistats/aistats2003.html
[40] C. M. Bishop, *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Secaucus, NJ, USA: Springer, 2006.
[41] G. McLachlan and T. Krishnan, *The EM Algorithm and Extensions*, vol. 382. New York, NY, USA: Wiley, 2007.
[42] C. J. Wu, "On the convergence properties of the EM algorithm," *Ann. Stat.*, vol. 11, pp. 95–103, 1983.
[43] S. Rangan, P. Schniter, E. Riegler, A. Fletcher, and V. Cevher, "Fixed points of generalized approximate message passing with arbitrary matrices," in *Proc. IEEE Int. Symp. Inf. Theory*, 2013, pp. 664–668.
[44] D. Wipf and S. Nagarajan, "Iterative reweighted ℓ_1 and ℓ_2 methods for finding sparse solutions," *IEEE J. Sel. Top. Signal Process.*, vol. 4, no. 2, pp. 317–329, Apr. 2010.
[45] Z. Zhang and B. D. Rao, "Sparse signal recovery in the presence of correlated multiple measurement vectors," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process.*, 2010, pp. 3986–3989.
[46] F. R. Kschischang, B. J. Frey, and H.-A. Loeliger, "Factor graphs and the sum-product algorithm," *IEEE Trans. Inf. Theory*, vol. 47, no. 2, pp. 498–519, Feb. 2001.

Maher Al-Shoukairi (S'17) received the B.Sc. degree in electrical engineering from the University of Jordan, Amman, Jordan, in 2005. He received the M.S. degree in electrical engineering from Texas A&M University, College Station, TX, USA, in 2008, after which he joined Qualcomm Inc. on the same year to date. He is currently working toward the Ph.D. degree in electrical engineering at the University of California at San Diego, San Diego, CA, USA.

Philip Schniter (S'92–M'93–SM'05–F'14) received the B.S. and M.S. degrees in electrical engineering from the University of Illinois at Urbana-Champaign, Champaign, IL,USA, in 1992 and 1993, respectively, and the Ph.D. degree in electrical engineering from Cornell University in Ithaca, NY, USA, in 2000. After receiving the Ph.D. degree, he joined the Department of Electrical and Computer Engineering, The Ohio State University, Columbus, OH, USA, where he is currently a Professor. From 1993 to 1996, he was a Systems Engineer at Tektronix Inc. in Beaverton, OR, USA. During 2008–2009, he was a Visiting Professor at Eurecom, Sophia Antipolis, France, and Supelec, Gif-sur-Yvette, France. During 2016–2017, he was a Visiting Professor at Duke University, Durham, NC, USA. His research interest include signal processing, wireless communications, and machine learning.

Bhaskar D. Rao (F'00) received the B.Tech. degree in electronics and electrical communication engineering from the Indian Institute of Technology of Kharagpur, Kharagpur, India, in 1979, and the M.S. and Ph.D. degrees from the University of Southern California, Los Angeles, CA, USA, in 1981 and 1983, respectively. Since 1983, he has been at the University of California at San Diego, San Diego, CA, USA, where he is currently a Distinguished Professor in the Department of Electrical and Computer Engineering. He is the holder of the Ericsson Endowed Chair in Wireless Access Networks and was the Director of the Center for Wireless Communications (2008–2011). His research interests include digital signal processing, estimation theory, and optimization theory, with applications to digital communications, speech signal processing, and biomedical signal processing. He was a Fellow of IEEE in 2000 for his contributions to the statistical analysis of subspace algorithms for harmonic retrieval. His received several paper awards: 2013 Best Paper Award at the Fall 2013, IEEE Vehicular Technology Conference for the paper "Multicell Random Beamforming with CDF-based Scheduling: Exact Rate and Scaling Laws," by Yichao Huang and Bhaskar D Rao, 2012 Signal Processing Society best paper award for the paper "An Empirical Bayesian Strategy for Solving the Simultaneous Sparse Approximation Problem," by D. P. Wipf and B. D. Rao published in IEEE TRANSACTION ON SIGNAL PROCESSING, vol. 55, no. 7, July 2007, 2008 Stephen O. Rice Prize paper award in the field of communication systems for the paper "Network Duality for Multiuser MIMO Beamforming Networks and Applications," by B. Song, R. L. Cruz, and B. D. Rao that appeared in the IEEE TRANSACTIONS ON COMMUNICATIONS, vol. 55, no. 3, pp. 618–630, March 2007. (http://www.comsoc.org/awards/rice.html), among others. He also received the 2016 IEEE Signal Processing Society Technical Achievement Award.