Unsourced Random Access with Coded Compressed Sensing: Integrating AMP and Belief Propagation

†Vamsi K. Amalladinne, *Member, IEEE*, ¶Asit Kumar Pradhan, *Member, IEEE*,
§Cynthia Rush, *Member, IEEE*, ‡Jean-Francois Chamberland, *Senior Member, IEEE*,
‡Krishna R. Narayanan, *Fellow, IEEE*
†Qualcomm Technologies, Inc.
¶Department of Electrical and Computer Engineering, University of Arizona
‡Department of Electrical and Computer Engineering, Texas A&M University
§Department of Statistics, Columbia University

***Abstract*—Sparse regression codes with approximate message passing (AMP) decoding have gained much attention in recent times. The concepts underlying this coding scheme extend to unsourced random access with coded compressed sensing (CCS), as first demonstrated by Fengler, Jung, and Caire. Specifically, their approach employs a concatenated coding framework with an inner AMP decoder followed by an outer tree decoder. In their original implementation, these two components work independently of each other, with the tree decoder acting on the static output of the AMP decoder. This article introduces a novel framework where the inner AMP decoder and the outer decoder operate in tandem, dynamically passing information back and forth to take full advantage of the underlying CCS structure. This scheme necessitates the redesign of the outer code as to enable belief propagation in a computationally tractable manner. The enhanced architecture exhibits significant performance benefits over a range of system parameters. The error performance of the proposed scheme can be accurately predicted through a set of equations known as state evolution of AMP. These findings are supported both analytically and through numerical methods.**

***Index Terms*—Unsourced random access, sparse regression codes, approximate message passing, belief propagation, coded compressed sensing, concatenated coding.**

I. Introduction and Background

Unsourced random access is a novel communication paradigm envisioned to accommodate the increasing traffic demands and heterogeneity of next generation wireless networks. This framework garnered significant research interest owing to the emergence of Internet of Things (IoT) and machine-driven communications. This model differs from the conventional multiple access paradigm in a number of ways. Conventional multiple access schemes are designed primarily for human-centric communications with sustained connections wherein the cost of coordination can be amortized over a long time period. However, this strategy may not be suitable for machine-centric communications because device transmissions are often sporadic with very short payloads. This new reality invites the creation of protocols in which it is not mandatory for active devices to reveal their identities. Rather, decoding is done only up to a permutation of the transmitted payloads, without regard for the identities of transmitting devices. Active devices wishing to reveal themselves can embed such information in their payloads. This approach enables all active devices to share a common codebook for their transmissions.

The unsourced random access channel (URA) is studied in [1], along with a random coding achievability bound for its capacity in the absence of complexity constraints. Subsequently, several practical coding schemes that aim to perform close to this conceptual benchmark have been proposed in the literature [2]–[10]. These schemes can be broadly divided into two categories: schemes that are built on conventional channel codes like LDPC or polar codes [3], [6], [9], [10]; and those that offer compressed sensing (CS) based solutions [4], [5], [7], [8]. Within this context, Amalladinne et al. [4] put forth a concatenated coding scheme that uses an inner CS code and an outer tree code. In doing so, their scheme takes advantage of the connection between the equivalence of unsourced multiple access and support recovery in high dimensional compressed sensing. Indeed, URA can be cast as a compressed sensing problem, albeit one whose excessive size precludes the straightforward application of existing solutions. Accordingly, Amalladinne et al. leverage a divide-and-conquer approach to split the information messages of active devices into several blocks, each amenable to standard CS solvers. Redundancy is employed in the form of an outer code to bind the

This material is based upon work supported, in part, by the National Science Foundation (NSF) under Grants CCF-1619085, CCF-1849883 & CCF-2131106, and by Qualcomm Technologies, Inc., through their University Relations Program. This work was presented in part at the International Symposium on Information Theory, 2020.

V. K. Amalladinne is with Qualcomm Research, Qualcomm Technologies, Inc., San Diego, CA, 92121, USA (e-mail: vamsia@qti.qualcomm.com). A. K. Pradhan is with the Department of Electrical and Computer Engineering, University of Arizona, Tucson, AZ 85721, USA (email: asitpradhan@arizona.edu). C. Rush is with the Department of Statistics, Columbia University, New York, NY 10027, USA. (email: cynthia.rush@columbia.edu). J.-F. Chamberland, and K. R. Narayanan are with the Department of Electrical and Computer Engineering, Texas A&M University, College Station, TX 77843, USA (emails: {chmbrlnd,krn}@tamu.edu).

information blocks that correspond to one message together. The performance of one such scheme is studied extensively in [4], and this revealed the natural tradeoff between error performance and computational complexity afforded by the allocation of parity check bits across blocks. This scheme, termed coded compressed sensing (CCS), employs the parity-check bits added in the encoding phase solely for the purpose of stitching. Yet, it turns out that the inner and outer decoding steps in CCS can be executed concurrently, and the redundancy employed during the transmission phase can be utilized to curtail the realm of possibilities for parity-check bits in subsequent stages [8]. This algorithmic improvement developed for CCS offers significant benefits both in terms of error performance and computational complexity. This enhancement also engenders new parameter allocation strategies beyond those observed in the CCS framework, some of which are studied in [8]. We refer the reader to [4], [8] for more details regarding this line of work. A motivation for this article is the realization that similar notions can be applied to other schemes related to URA and beyond.

Approximate message passing (AMP) [11]–[13] refers to a broad class of iterative algorithms derived from message passing algorithms on dense factor graphs; this approach has demonstrated good performance while acting as decoders in the context of wireless communications. An early instance of AMP applied to digital communication pertains to the efficient decoding of a sparse regression code (SPARC) [14]–[17]. SPARCs are a coding scheme based on high-dimensional linear regression introduced for the single-user AWGN channel by Barron and Joseph [18], [19]. It was rigorously proved in [15] that SPARCs with an AMP decoder and an appropriate power allocation achieves the capacity of a single-user AWGN channel asymptotically and, subsequently, there has been much effort to improve the finite block length performance of single user SPARC [20]–[26] by leveraging standard techniques like spatial coupling, concatenated coding, or higher order modulations. The performance improvements demonstrated by these techniques provide a strong incentive to explore similar frameworks in the context of unsourced multiple access; we discuss one such approach in this article.

The first application of an AMP decoder to the URA setting is due to Fengler, Jung, and Caire [5]. Therein, the authors draw a connection between the structure of CCS and SPARC constructions. They extend the CCS framework [4] by using a design (measurement) matrix that does not assume a block diagonal structure, and they apply AMP as part of the message recovery process. They rely on the tree decoder proposed in [4] to accomplish the stitching process, once individual messages have been reconstructed by the AMP decoder. Also, they show that this concatenated construction asymptotically achieves the symmetric sum rate Shannon capacity of the MAC channel, thereby extending the optimality results of SPARC in [15] to scenarios with multiple users. In this article, we build on the insights developed in [8] to devise novel message passing rules that integrate the outer code and AMP. The main contributions of this article are as follows.

1) A novel framework that facilitates dynamic interactions between inner and outer decoding components of the concatenated coding structure introduced in [5] is proposed for the unsourced random access problem.
2) We show that the redundancy intrinsic to the outer code can be harnessed to assist the convergence of AMP.
3) We develop a modified outer code that allows passing meaningful messages back and forth between the inner AMP and the outer tree decoding components. This revised tree code enables us to employ fast Fourier transform (FFT) techniques to conduct belief propagation in a computationally efficient manner. The reader may notice a similarity between our framework and the way messages are passed in non-binary LDPC codes.
4) We provide finite-block length simulation results for this proposed scheme to demonstrate the performance gain it offers over previously published schemes for scenarios of interest [4], [5].
5) We develop a framework to track the state evolution and find that it accurately predicts the error performance of the proposed concatenated coding scheme. Also, within the proposed framework, we prove that the state evolution is accurate.

Concatenated schemes that combine AMP with an outer code have been proposed in [20], [27] to improve the finite block length performance in the context of single user SPARC. In [20], a high rate LDPC code is used to protect sections of AMP with less allocated power. The decoding operation involves three steps: the first step uses AMP decoding, the second step decodes the outer LDPC codes by employing the soft outputs produced by AMP in the first stage and the third step re-runs the AMP decoder after removing the contribution of successfully decoded sections from the second stage. This approach results in a steep waterfall in error performance; a phenomenon that is not achieved by standalone AMP decoder for small block lengths. However, this architecture does not allow dynamic interactions between AMP and the LDPC decoder. In other words, the LDPC decoder does not assist the convergence of AMP and it only works on soft outputs produced by AMP upon convergence. Compressed-coding is proposed and analyzed in [27], where AMP is combined with an outer generic forward error correction (FEC) code. Therein, the authors show that with a careful design of the underlying FEC code, the compressed-coding scheme can achieve the single-user Gaussian capacity asymptotically. Yet, their analysis hinges on the assumption that the state evolution for AMP remains accurate even in the presence of an outer FEC decoder. In this article, we do not rely on such assumptions; rather, we prove that the state evolution for AMP is indeed accurate under certain conditions for the proposed architecture. It is also worth noting that the schemes in [20], [27] do not admit straightforward extensions to the URA paradigm, which points to the novel and unique character of our contributions.

A. Organization

The remainder of this article is organized as follows. In Section II, we describe the system model and the broad CCS-AMP architecture. We also introduce the reader to the relevant notation used throughout this paper. In Section III, we

provide a detailed description of the revised outer code and we outline the key distinguishing factors between our revised outer code and the one developed in [4]. We then introduce a framework that allows soft decoding of the outer code, which can be employed in tandem with the decoding of the inner AMP code. A fast implementation of the above algorithm, which leverages FFT techniques, is also detailed in this part. Section IV focuses on the inner code and the AMP decoder. Therein, we explain how the structure of underlying outer code can be harnessed to assist the convergence of AMP through a novel denoising function. Furthermore, we provide guiding principles to design good outer codes which serve the purpose of assisting AMP convergence, as well as stitching individual messages together once AMP has converged. Finally, for the proposed scheme, we describe a framework to track the state evolution, a tool used to predict the error performance of AMP decoders. Simulation results are reported in Section V, and conclusions are drawn in Section VI.

II. System Model and Conceptual Framework

Consider a situation where $K_{\rm a}$ active devices out of a population of $K_{\rm tot}$ devices ($K_{\rm a} \ll K_{\rm tot}$) each wish to send a message to an access point. Without loss of generality, we label these active devices using integers from one to $K_{\rm a}$. The transmission process takes place over a multiple access channel, with a time duration of n channel uses (real degrees of freedom). We denote the message of device i by $\mathbf{w}_i$, and we use $\mathbf{x}_i$ to represent the corresponding transmitted waveform. The signal received at the destination takes the form

$$\mathbf{y} = \sum_{i=1}^{K_{\rm a}} \mathbf{x}_i + \mathbf{z} \tag{1}$$

where $\mathbf{x}_i \in \mathbb{R}^n$. The additive noise component $\mathbf{z}$ is composed of a sequence of independent Gaussian elements, each with distribution $\mathcal{N}(0,1)$. All the devices share a common codebook $\mathcal{C}$, as is customary in unsourced random access. Consequently, $\mathbf{x}_i$ is a function of payload $\mathbf{w}_i$, but not of the identity of device i itself. For ease of exposition, we assume that a frame synchronization beacon or an alternate mechanism enables coherent transmission, although we note that this framework can be extended to more general settings [28].

The access point is tasked with producing an unordered list $\widehat{W}(\mathbf{y})$ of candidate messages based on the received signal $\mathbf{y}$. The size of the output list is constrained and cannot exceed $K_{\rm a}$, i.e., $\big|\widehat{W}(\mathbf{y})\big| \leq K_{\rm a}$. The performance of this communication scheme is assessed using the per-user probability of error (PUPE), which is the predominant evaluation criterion for unsourced random access [1]. Mathematically, we have

$$P_{\rm e} = \frac{1}{K_{\rm a}} \sum_{i=1}^{K_{\rm a}} \Pr\Big(\mathbf{w}_i \notin \widehat{W}(\mathbf{y})\Big) \tag{2}$$

where $\mathbf{w}_i$ is the payload of active device i. Our prime design goal consists of creating a pragmatic, low-complexity scheme that enables the communication of $K_{\rm a}$ messages to the access point under the URA paradigm with a probability of failure $P_{\rm e} \leq \epsilon$. Furthermore, we wish to do so utilizing a small number of channel uses ($n \approx 30,000$), and with devices expending as little energy as possible.

Having established a framework and a task, we are ready to initiate our treatment of CCS-AMP. We begin with a brief overview of the proposed architecture. We then discuss the specifics of an outer code tailored to its integration with the AMP framework. We finish the treatment of our new scheme with the description and analysis of the enhanced AMP algorithm.

A. CCS-AMP Architecture

The selection of a codeword by an active device follows the general structure obtained by combining an outer code akin to Amalladinne et al. [29] and the SPARC-like inner encoding of Fengler et al. [5]. Specifically, consider a payload $\mathbf{w} \in \{0,1\}^w$. Redundancy is first added to this message in the form of an outer code. Explicitly, this w-bit binary message $\mathbf{w}$ is partitioned into L blocks, where the ℓth fragment contains w_ℓ information bits and $\sum_{\ell=1}^{L} w_\ell = w$. In our treatment of outer codes, we frequently represent message $\mathbf{w}$ as a concatenation of fragments:

$$\mathbf{w} = \mathbf{w}(1)\mathbf{w}(2)\cdots\mathbf{w}(L). \tag{3}$$

The outer encoder appends p_ℓ parity check bits to fragment $\mathbf{w}(\ell)$, bringing the total length of block $\mathbf{v}(\ell) = \mathbf{w}(\ell)\mathbf{p}(\ell)$ to $v_\ell = w_\ell + p_\ell$. The structure of the message produced by the outer code assumes the form

$$\mathbf{v} = \underbrace{\mathbf{w}(1)}_{\mathbf{v}(1)}\ \underbrace{\mathbf{w}(2)\mathbf{p}(2)}_{\mathbf{v}(2)}\cdots\underbrace{\mathbf{w}(L)\mathbf{p}(L)}_{\mathbf{v}(L)}. \tag{4}$$

Equivalently, vector $\mathbf{v}$ can be viewed as containing L disjoint blocks: $\mathbf{w}(1), \mathbf{w}(2)\mathbf{p}(2), \ldots, \mathbf{w}(L)\mathbf{p}(L)$. For the sake of uniformity, we regard $\mathbf{p}(1)$ as a parity segment of length zero, i.e., $p_1 = 0$.

The original tree code found in [5], [29] includes mixed blocks containing a combination of information and parity-check bits, as in (4). In our revised implementation, we focus on homogeneous blocks: every coded block features either information bits or parity-check bits, but not a combination of both. Thus, we have either $w_\ell = 0$ or $p_\ell = 0$ for all $\ell \in [1:L]$. As we will see shortly, this structure eliminates certain dependencies among blocks and it enables the exploitation of a circular convolution structure propitious to the application of FFT methods during decoding. Such a design naturally leads to a partition of $[1:L]$ into the set of information blocks $\mathcal{W}$ and the collection of parity blocks $\mathcal{P} = [1:L] \setminus \mathcal{W}$. The parity-check bits contained in $\mathbf{p}(\ell)$, $\ell \in \mathcal{P}$, act as constraints on the bits coming from other blocks.

As part of the next encoding step, every block is turned into a message index of length $m_\ell = 2^{v_\ell}$. This action is emblematic of CCS and, therefore, it is worth going over carefully. Conceptually, the message index is

$$\mathbf{m}(\ell) = f_{\mathbb{F}_2^{v_\ell} \to \{0,1\}^{m_\ell}}(\mathbf{v}(\ell)), \tag{5}$$

where the function $f_{\mathbb{F}_2^{v_\ell} \to \{0,1\}^{m_\ell}}(\mathbf{v}(\ell))$ can be grasped by regarding argument $\mathbf{v}(\ell)$ as an integer in binary form. The output is a length-2^{v_ℓ} real vector with zeros everywhere,

except for a one at location $[\mathbf{v}(\ell)]_2$. The shorthand notation $[\cdot]_2$ designates an integer expressed using a radix of 2 (possibly with leading zeros), and the location indexing of entries in $\mathbf{m}(\ell)$ ranges from zero to $2^{v_\ell}-1$. For example, if $\mathbf{v}(\ell)=101$ then $v_\ell=3$, $m_\ell=8$, and $\mathbf{m}(\ell)=00000100$ because $[\mathbf{v}(\ell)]_2=5$. Message $\mathbf{m}$ is subsequently created by concatenating individual sections,

$$\begin{aligned}\mathbf{m} &= \mathbf{m}(1)\cdots\mathbf{m}(L)\\ &= f_{\mathbb{F}_2^{v_1}\to\{0,1\}^{m_1}}(\mathbf{v}(1))\cdots f_{\mathbb{F}_2^{v_L}\to\{0,1\}^{m_L}}(\mathbf{v}(L)).\end{aligned} \tag{6}$$

As we will see, section sizes are typically very large and, accordingly, every block can be viewed as being one-sparse. The induced vector is reminiscent of a SPARC codeword [14], [18], [19]. The structure in (6) is slightly more general than the form in [5] because it admits the possibility of having sections of different sizes. The resemblance is nevertheless manifest.

Next, we describe how signal $\mathbf{x}$ is generated from $\mathbf{m}$. Let $\mathbf{A}$ be an $n\times m$ matrix over the real numbers, where $m=\sum_{\ell=1}^{L} m_\ell$; and let $\mathbf{D}$ be diagonal matrix with non-negative entries. Transmitted signals in $\mathcal{C}$ are created via the product $\mathbf{x}=\mathbf{ADm}$ over the field of real numbers. Matrix $\mathbf{D}$ accounts for the power allocated to every section and, accordingly, diagonal entries are constant within each block. The amplitude of the signal for section ℓ is denoted by d_ℓ. Given that all active devices utilize a same codebook, this process yields a received vector $\mathbf{y}$ of the form

$$\mathbf{y}=\sum_{i=1}^{K_\mathrm{a}}\mathbf{ADm}_i+\mathbf{z}=\mathbf{AD}\underbrace{\left(\sum_{i=1}^{K_\mathrm{a}}\mathbf{m}_i\right)}_{\mathbf{s}}+\mathbf{z}=\mathbf{ADs}+\mathbf{z} \tag{7}$$

where $\mathbf{s}$ is such that all its sections, $\mathbf{s}(\ell)=\sum_{i=1}^{K_\mathrm{a}}\mathbf{m}_i(\ell)$ with $\ell\in[1:L]$, are K_a sparse.[1] Matrix $\mathbf{A}$ is normalized in that the 2-norm of every column is equal to one. The interpretation of (7) as a SPARC-like model, originally put forth in [5], immediately extends to the present case. Moreover, as in [5], the resulting multiple access channel can be viewed as the combination of a point-to-point channel $\mathbf{s}\to\mathbf{ADs}+\mathbf{z}$ and an outer binary adder MAC $\mathbf{s}=\sum_{i=1}^{K_\mathrm{a}}\mathbf{m}_i$. Therein, the authors refer to these components as the *inner* and *outer channels*, respectively. They also draw a distinction between the *inner* and *outer encoder/decoder* pairs.

Our article embraces the aforementioned categorization for the channel components, *inner* and *outer channels*. However, we do not subscribe to the latter dissociation between the decoders. Rather, we seek to exploit the fact that decoding can be improved if information is allowed to flow dynamically between the inner and outer channel components while iterative decoding takes place. As mentioned above, the impetus behind our approach stems from a potential algorithmic enhancement that was first noticed in the context of coded compressed sensing [8]. Therein, the authors show how decoded blocks in earlier stages can inform the CS recovery process at subsequent stages through tracking sets of admissible parity patterns. Indeed, the collection of all active paths in the tree decoder at a particular stage dictates the set of permissible parity patterns at the subsequent stage. This information can be leveraged to reduce the difficulty of the support recovery task at the later stages and concurrently improve performance. Additional details regarding this algorithmic improvement for CCS can be found in [8].

The relationship between parity bits in the outer code and support recovery in the inner channel is more subtle in the present context, which we call CCS-AMP. There are two architectural aspects that can help guide the convergence of our iterative decoding process. The inherent block sparsity contained in $\mathbf{s}$ provides a foundation for the AMP denoiser [5]. Moreover, there is an embedded factor graph structure in the tree code that, when designed carefully, is amenable to belief propagation. Thus, within each iteration of the AMP algorithm, the state estimates can be updated via message passing on the factor graph induced by the tree code. Under this novel approach, as the ambiguity on some sections diminishes, the uncertainty on the belief states of their (graph) neighbors also decreases. Incorporating these two modalities within the iterative decoding process offers significant performance benefits, beyond the concentration afforded by the AMP algorithm alone. We will see shortly how these pieces of information can be integrated into a consolidated iterative scheme. Admittedly, in the actual decoding process, the interactions between the tree decoder and the AMP algorithm are more complex than described above. Nevertheless, this simplified characterization strongly hints at an opportunity to improve performance by running AMP and the tree decoder in tandem, dynamically passing information back and forth between these two components. The specifics of our implementation and how it facilitates such dynamic exchanges of information are contained in the upcoming sections.

B. Notation

Notation in this article is heavy despite our best effort to keep it to a minimum. We therefore offer a brief overview of the notation we adopt, along with a table summary, for the sake of readability. A strong motivation for Table I stems from the fact that we frequently transition between equivalent representations to introduce concepts and explain algorithms. An outer codeword is a sequence of bits of the form $\mathbf{v}=\mathbf{v}(1)\cdots\mathbf{v}(L)$. The number of bits in $\mathbf{v}(\ell)$ is v_ℓ. The value k_ℓ of binary block $\mathbf{v}(\ell)$ is obtained by regarding the payload as an integer expressed in radix-2, which we write as $k_\ell=[\mathbf{v}(\ell)]_2$. The block index for $\mathbf{v}(\ell)$ is a standard basis element with a one at location k_ℓ and zeros everywhere else, i.e., the entries of $\mathbf{m}(\ell)$ are given by $\mathbf{m}(\ell,k)=\delta_{k_\ell}(k)$ where $\delta_{k_\ell}(k)=1$ when $k=k_\ell$ and $\delta_{k_\ell}(k)=0$ otherwise. The sparse vector $\mathbf{m}$ is obtained by concatenating the index messages, $\mathbf{m}=\mathbf{m}(1)\cdots\mathbf{m}(L)$. The integer vector representation of this same message is $\mathbf{k}=(k_1,\ldots,k_L)$. The message produced by device i is annotated with either a subscript, $\mathbf{v}_i$ or $\mathbf{m}_i$, or a superscript $\mathbf{k}^{(i)}$. The aggregate signal is equal to $\mathbf{s}=\sum_{i=1}^{K_\mathrm{a}}\mathbf{m}_i$. There is no equivalent representation for $\mathbf{s}$ in compact form or integer form. Consequently, we resort to collections of messages

[1] Technically, every section is at most K_a-sparse; when $m_\ell\gg K_\mathrm{a}$, improbable index collisions may occasionally reduce the number of non-zero entries to less than K_a.

wherever needed. We define sections of $\mathbf{s}$ through the sum $\mathbf{s}(\ell) = \sum_{i=1}^{K_\mathrm{a}} \mathbf{m}_i(\ell)$, and we employ $\mathbf{s}(\ell, k)$ to refer to the kth entry of its ℓth section. We extend this convention to all the vectors that admit a fragmented representation.

III. Outer Code Revisited

This section focuses on the outer code, which takes the form of a low-density parity-check (LDPC) code. An important distinction between the original CCS framework and CCS-AMP from an outer code perspective comes from the fact that the former applies consistency checks sequentially to short lists on the order of K_a items, whereas CCS-AMP produces large belief vectors where complete blocks are processed to update estimates. To accommodate this new reality, the outer code employed in this article differs from the original tree code introduced in [4], [29]. Below, we describe the revised encoding process for our system and how it deviates from the original tree code incarnation. We also introduce a soft decoder for our new outer code suitable for dynamic interactions with the inner code and AMP.

A. Alternate Outer Encoding

In a manner akin to the original CCS scheme, the construction of our outer code starts by partitioning information bits into fragments. Parity patterns are then added to blocks in a causal fashion, leading to vector $\mathbf{v}$. To this extent, the alternate outer code subscribes to the same structure as the original one found in [29]. However, in our revised construction, parity bits are created differently. For $\ell \in \mathcal{P}$, we use $\mathcal{W}_\ell$ to denote the collection of blocks on which parity block $\mathbf{v}(\ell)$ operates. Every parity block is obtained using the following three-step sequence. We first take a random linear combination of the bits in each fragment of $\mathcal{W}_\ell$; specifically, $\mathbf{v}(j)\mathbf{G}_{j,\ell}$ for $j \in \mathcal{W}_\ell$, where $\mathbf{G}_{j,\ell}$ is selected at random from $\{0,1\}^{v_j \times v_\ell}$. Within this step, vector operations are taken over Galois field $\mathbb{F}_2$. We transpose these combinations from the space of length-v_ℓ binary vectors over $\mathbb{F}_2$ to the ring of integers modulo-2^{v_ℓ}, which we denote by $\mathbb{Z}/2^{v_\ell}\mathbb{Z}$. We then add the resulting elements, which we call parity precursors, within the ring of integers modulo-2^{v_ℓ}. Finally, we convert the ensuing sum back to a vector in $\mathbb{F}_2^{v_\ell}$. The sequence of bits obtained through this process determines parity block $\mathbf{v}(\ell)$. Mathematically, these operations can be expressed as

$$\begin{aligned}\mathbf{v}(\ell) &\equiv \sum_{j\in\mathcal{W}_\ell} [\mathbf{v}(j)\mathbf{G}_{j,\ell}]_{\mathbb{Z}/2^{v_\ell}\mathbb{Z}} \\ &\equiv \sum_{j\in\mathcal{W}_\ell} [\mathbf{v}(j)\mathbf{G}_{j,\ell}]_{\mathbb{Z}} \mod 2^{v_\ell}.\end{aligned} \tag{8}$$

The notation $[\cdot]_{\mathbb{Z}/2^{v_\ell}\mathbb{Z}}$ emphasizes that the argument is interpreted as an element of the quotient ring $\mathbb{Z}/2^{v_\ell}\mathbb{Z}$, and the equivalence relation '$\equiv$' denotes equality in $\mathbb{Z}/2^{v_\ell}\mathbb{Z}$. Equivalently, one can think of these operations as taking the module-2^{v_ℓ} sum of the parity precursors. When the context is unambiguous, we abbreviate such terms by retaining the square brackets, but omitting the explicit ring subscript. In such cases, we maintain the use of '$\equiv$' to designate a congruence relation. Conceptually, $\mathbf{v}(\ell)$ contains non-linear constraints on the information bits from fragments in $\mathcal{W}_\ell$. It is worth mentioning that the parity encoding process in (8) is more contrived than the random linear combinations utilized in the original tree code [29]. The benefit of this more intricate encoding process is that it induces a cyclic structure on parity precursors conducive to the circular convolution, which is befitting to the eventual application of FFT techniques.[2] The advantages of this construction will be revealed shortly, when we pair the outer decoding with the AMP inner decoder. For the time being, we demonstrate that the encoding in (8) shares several attributes with the original tree code.

The outer code construction admits a factor graph representation [30]. Our upcoming discussion relies heavily on this abstraction and, as such, we elaborate on this analogy. We denote the variable nodes by $\{s_\ell : \ell \in 1, \dots L\}$, and we label the factors using $\{a_p : p \in \mathcal{P}\}$. By design, an edge necessarily exists between a_p and s_p. In addition, an edge is placed between s_j and a_p whenever $j \in \mathcal{W}_p$. The factor associated with a_p, where $p \in \mathcal{P}$, is derived from the computation of the corresponding parity block and local observations. Given $p \in \mathcal{P}$, we emphasize that candidate sections $(\hat{\mathbf{v}}(j) : j \in \mathcal{W}_p \cup p)$ can only come from the same message if they are collectively parity consistent. We introduce indicator function $\mathcal{G}_{a_p}(\cdot)$ to assess local consistency,

$$\begin{aligned}&\mathcal{G}_{a_p}\left(\hat{\mathbf{v}}(j) : j \in \mathcal{W}_p \cup p\right) \\ &\quad = \mathbf{1}\left(\sum_{j\in\mathcal{W}_p} [\hat{\mathbf{v}}(j)\mathbf{G}_{j,p}] \equiv \hat{\mathbf{v}}(p)\right).\end{aligned} \tag{9}$$

This function returns one when the block indices in its argument are parity consistent with respect to (8), and it outputs zero otherwise. As a side note, we adopt an overloaded notation for factor $\mathcal{G}_{a_p}(\cdot)$ in that we employ the same notation for all the message representations listed in Table I although, technically, these functions have different domains. Since the overloaded functions have equivalent meanings, this should not lead to any confusion, yet it simplifies exposition.

Using factor graph notation, the neighbors of factor a_p, written $N(a_p)$, are the elements of $\mathcal{W}_p \cup \{p\}$. Moreover, the graph neighbors of variable s_ℓ, written $N(s_\ell)$, are the factors associated with the parity equations where $\mathbf{v}(\ell)$ appears either as a summand or as the parity. We offer an elementary example to help cement these notions. Consider an outer code where $\mathbf{v}(1), \mathbf{v}(2), \mathbf{v}(4)$ are information blocks, and parity blocks are given by

$$\mathbf{v}(3) \equiv \left[\mathbf{v}(1)\mathbf{G}_{1,3}\right] + \left[\mathbf{v}(2)\mathbf{G}_{2,3}\right] \tag{10}$$

$$\mathbf{v}(5) \equiv \left[\mathbf{v}(1)\mathbf{G}_{1,5}\right] + \left[\mathbf{v}(2)\mathbf{G}_{2,5}\right] + \left[\mathbf{v}(4)\mathbf{G}_{4,5}\right] \tag{11}$$

Then, $N(s_1) = N(s_2) = \{a_3, a_5\}$, $N(s_3) = \{a_3\}$, and $N(s_4) = N(s_5) = \{a_5\}$; likewise, $N(a_3) = \{s_1, s_2, s_3\}$ and $N(a_5) = \{s_1, s_2, s_4, s_5\}$. The ensuing factor graph, which

[2] This strategy allows us to employ fast transform techniques to dynamically pass information between the outer and inner decoders. The reader may notice that these binary vectors can be mapped to elements of the finite field $\mathbf{GF}(2^{v_\ell})$ instead of elements of the quotient ring $\mathbb{Z}/2^{v_\ell}\mathbb{Z}$. These two representations exhibit similar structural properties under the proposed framework, and we choose to adhere to the ring notation throughout this article.

Representations			
Objects	Compact	Integer	Sparse Vector
Block	$\mathbf{v}(\ell) \in \{0,1\}^{v_\ell}$	$k_\ell = [\mathbf{v}(\ell)]_2$	$\mathbf{m}(\ell) : \mathbf{m}(\ell,k) = \delta_{k_\ell}(k)$
Message	$\mathbf{v} = \mathbf{v}(1)\cdots\mathbf{v}(L)$	$\mathbf{k} = (k_1,\ldots,k_L)$	$\mathbf{m} = \mathbf{m}(1)\cdots\mathbf{m}(L)$
Signal	$\{\mathbf{v}_i : i \in [K_\mathrm{a}]\}$	$\{\mathbf{k}^{(i)} : i \in [K_\mathrm{a}]\}$	$\mathbf{s} = \sum_i \mathbf{m}_i$

TABLE I
SUMMARY OF VARIOUS REPRESENTATIONS.

captures parity consistencies among blocks, appears in Fig. 1.

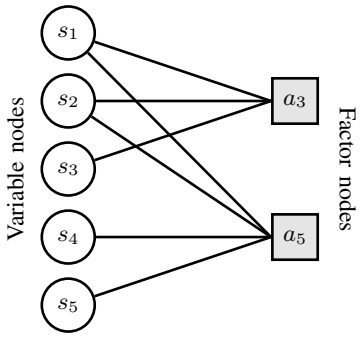

Fig. 1. In the factor graph interpretation of the outer code, every block yields a variable node, $\{s_\ell : \ell \in 1,\ldots,L\}$, and every parity equation produces a factor, $\{a_p : p \in \mathcal{P}\}$.

B. Validation of Candidate Codewords

As mentioned above, all active devices use a common encoding scheme and they collectively produce vectors $\mathbf{v}_1,\ldots,\mathbf{v}_{K_\mathrm{a}}$. The bijection between $\mathbf{v}$ and $\mathbf{m}$ defined in (6) naturally introduces a correspondence between $\mathbf{s}(\ell)$ and subset $\{\mathbf{v}_1(\ell),\ldots,\mathbf{v}_{K_\mathrm{a}}(\ell)\}$. Specifically, $\mathbf{v}(\ell)$ is contained in the latter set whenever $\mathbf{s}(\ell)$ has a value of one (or greater) at index location $[\mathbf{v}(\ell)]_2$. We recall that the original tree decoder (with hard decisions) operates on $\mathbf{s}$ and it is tasked with recovering the collection $\{\mathbf{v}_1,\ldots,\mathbf{v}_{K_\mathrm{a}}\}$. That is, it must stitch the segments of valid codewords together. At the onset of the process, the decoder extracts lists of blocks from $\mathbf{s}$, one for every level ℓ. Tentative codewords are formed by concatenating one fragment from each list, abiding by the natural progression from level 1 to level L. Consider a candidate codeword written as

$$\mathbf{v}_\mathrm{c} = \mathbf{w}_{i_1}(1)\mathbf{w}_{i_2}(2)\mathbf{p}_{i_2}(2)\cdots\mathbf{w}_{i_L}(L)\mathbf{p}_{i_L}(L), \qquad (12)$$

where i_ℓ is the index of the fragment employed to form $\mathbf{v}_\mathrm{c}$ at level ℓ. The decoder attempts to validate this candidate by recreating its parity patterns. For instance, it begins with information fragment $\mathbf{w}_{i_1}(1)$ and determines parity pattern $\mathbf{p}(2)$ using a condition akin to (8). If the resulting parity pattern matches $\mathbf{p}_{i_2}(2)$, the decoder proceeds forward to the next block; else candidate codeword $\mathbf{v}_\mathrm{c}$ is marked as invalid and it is discarded immediately. If $\mathbf{v}_\mathrm{c}$ gets through stage $(\ell-1)$, then the outer decoder continues by constructing $\mathbf{p}(\ell)$ using fragments $\mathbf{w}_{i_1}(1),\ldots,\mathbf{w}_{i_{\ell-1}}(\ell-1)$ and a parity condition akin to (8). Once again, the candidate codeword is retained if $\mathbf{p}(\ell)$ matches $\mathbf{p}_{i_\ell}(\ell)$; else it is dropped. Altogether, $\mathbf{v}_\mathrm{c}$ remains a codeword candidate for as long as parity blocks are consistent with the information fragments that precede them. To justify this procedure, it is relevant to recognize that $\mathbf{p}(\ell)$ is computed using fragments $\mathbf{w}_{i_1}(1),\ldots,\mathbf{w}_{i_{\ell-1}}(\ell-1)$; whereas $\mathbf{p}_{i_\ell}(\ell)$, coming from a valid codeword, was created by applying (8) to $\mathbf{w}_{i_\ell}(1),\ldots,\mathbf{w}_{i_\ell}(\ell-1)$. Thus, the ability of the outer decoder to detect invalid codewords is predicated on the probability that parity patterns generated through mismatched fragments $\mathbf{w}_{i_1}(1),\mathbf{w}_{i_2}(2),\ldots\mathbf{w}_{i_L}(L)$ correspond to the embedded parity patterns $\mathbf{p}_{i_2}(2),\ldots,\mathbf{p}_{i_L}(L)$. We stress that valid codewords whose fragments are on the lists are never discarded by the outer decoder because their parity patterns are necessarily self-consistent.

A detailed characterization of tree decoding (with hard decisions) and its expected performance can be found in [4] when parity bits are generated through random linear combinations, with the entries in $\mathbf{G}_{j,\ell} \in \{0,1\}^{w_j \times p_\ell}$ being independent Rademacher trials (Bernoulli trials with parameter half). The analysis therein relies on three key properties, which we enumerate below. Consider erroneous pseudo-codeword

$$\mathbf{v}_\mathrm{e} = \mathbf{w}_{i_1}(1)\mathbf{w}_{i_2}(2)\mathbf{p}_{i_2}(2)\cdots\mathbf{w}_{i_L}(L)\mathbf{p}_{i_L}(L), \qquad (13)$$

where indices $i_1, i_2, \ldots, i_L$ are not all from a same encoded message. Given that $\mathbf{v}_\mathrm{e}$ is invalid, the tree decoder is tasked with detecting and discarding it.

Remark 1. *In examining the probability that a tree decoder is successful in this singular endeavor and to assess average computational complexity, the following facts come into play.*

1) *The collection $\mathbf{p}_\mathrm{e}(\ell)$ of parity bits is either statistically discriminating*[3] *or, as a block, it is uninformative. In the former case, the probability that $\mathbf{p}_{i_\ell}(\ell)$ is consistent with pseudo-codeword $\mathbf{v}_\mathrm{e}$ is equal to 2^{-p_ℓ}.*
2) *The list of statistically discriminating parity blocks within $\mathbf{p}_{i_2}(2),\ldots,\mathbf{p}_{i_L}(L)$ depends on the index se-*

[3]Informally, given an erroneous sequence, *statistically discriminating* parity check bits each have a non-zero probability of revealing the mismatch, whereas *uninformative* parity check bits become ineffective at identifying the erroneous codeword due partly to the mismatched index sequence $i_1,\ldots,i_L$. In the latter case, these parity check bits are trivially met irrespective of the generator matrices and the parity equations. We refer the reader to Appendix A-D in [4] for a rigorous treatment of these concepts.

quence $i_1, \ldots, i_L$, with a re-entry to a previously visited level reducing the probability that the corresponding parity block remains statistically discriminating. For example, if $i_3 = i_1$, then we say that the sequence has re-entered level i_1 during stage 3.

3) *The conditional distribution on the collection of statistically discriminating blocks, given $i_1, \ldots, i_L$, is permutation invariant. That is, the precise state labeling $i_1, \ldots, i_L$ is unimportant; only the order in which previously visited states are re-entered matters (over the random ensemble of generating matrices). Mathematically, if $\pi(\cdot)$ is a permutation function on admissible indices, the probability that the tree decoder tags $\mathbf{v}_{\mathrm{e}}$ as invalid is the same as the probability that the decoder recognizes*

$$\mathbf{v}_{\mathrm{e}_\pi} = \mathbf{w}_{\pi(i_1)}(1)\mathbf{w}_{\pi(i_2)}(2)\mathbf{p}_{\pi(i_2)}(2)\cdots \mathbf{w}_{\pi(i_L)}(L)\mathbf{p}_{\pi(i_L)}(L) \tag{14}$$

as an erroneous codeword.

An important aspect of the novel encoding scheme introduced in (8) is the fact that it preserves the three properties listed above whenever the entries in $\mathbf{G}_{j,\ell} \in \{0,1\}^{v_j \times v_\ell}$ are independent Rademacher trials. Since the performance characterization of the tree code is based on expected behavior, the fact that the conditional probability of an erroneous codeword being detected remains the same under (8), given index sequence $i_1, \ldots, i_L$, is enough to ensure that findings derived for the original tree code extend to the new outer encoding process. In other words, the performance results presented in our previous work on tree codes are also valid in the current context. The third item enumerated above guarantees that the complexity reduction exposed in [4], and related to the Bell numbers, is present under the new encoding as well. This connection is pertinent because it offers performance guarantees for detection applied to the revised outer code, under certain conditions, thereby pointing to the practicality of our alternate approach for parity generation. Also, the last step of the overall CCS-AMP decoding process proposed in this work is essentially message disambiguation based on hard decision, as before. We formalize these findings below.

Theorem 2. *Under hard decision decoding and for identical factor graphs, the expected performance of the revised outer code based on* (8) *is equal to that of the original tree code found in [4], provided that the elements of $\{\mathbf{G}_{j,\ell}\}$ are i.i.d. Rademacher trials. The complexity of the decoding process in both settings is analogous, with the respective outer decoders validating candidate codewords by checking the consistency of parity patterns sequentially, starting from root fragments.*

Proof: To relate the expected performance of the alternate outer code to that of the original tree code, it suffices to check properties 1 and 2 in Remark 1. The complexity reduction statement relies on the third property therein. These three properties and, hence, Theorem 2 are established in Appendix A. ∎

C. Soft Outer Decoding

This section focuses on the soft decoding of the revised *outer code*, which can be employed in tandem with the iterative decoding of the *inner code*. The encoding process and, specifically, the parity generation defined in (8) induces a generalized Markov structure on the outer code. The AMP composite iterative algorithm, as we will see, utilizes two steps: the computation of a residual, and an update of the state estimate based on an effective observation. In its original form [5], the denoising step leverages solely the sparse structure of $\mathbf{s}$. Yet, the outer code imposes parity consistency conditions, beyond sparsity; the corresponding graphical structure can too inform the denoising step. To describe this algorithmic opportunity, it is useful to think of the soft outer decoder as getting an observation $\mathbf{r}$ of the form

$$\mathbf{r} = \mathbf{D}\mathbf{s} + \tau\boldsymbol{\zeta} \tag{15}$$

where $\boldsymbol{\zeta}$ is an i.i.d. $\mathcal{N}(0,1)$ random vector and τ is a scaling parameter that captures the standard deviation. Based on the underlying factor graph inherited from the outer code, this sub-component of the denoiser seeks to produce estimates for the elements of $\mathbf{s}$ using iterative message passing. Naively, the construction of $\mathbf{s}$ in (7) imposes a sparsity constraint within each block, whereas the generalized Markov structure described above captures dependencies between information and parity bits across blocks. In our proposed framework, sparsity is dictated primarily through the AMP iteration; whereas parity factors are leveraged within the denoiser via belief propagation. As a side note, we emphasize that the complexity of implementing an optimal state estimator for $\mathbf{s}$ based on effective observation $\mathbf{r}$ is often cost prohibitive for realistic designs. Nevertheless, it is possible to focus on local aspects of the factor graph associated with the outer code, computing beliefs using suitable message passing rules, extrinsic information, and applying factor functions derived from the parity generation mechanism of (8).

The message passing rules we utilize for the outer factor graph are presented below. We expound on the rationale behind them in Appendix B. The factor function associated with check nodes admits a product decomposition and it is given by

$$\mathcal{G}(\mathbf{k}) = \prod_{a_p \in \mathcal{P}} \mathcal{G}_{a_p}(\mathbf{k}_{a_p}) \tag{16}$$

where $\mathbf{k} = (k_\ell : \ell \in [L])$, $\mathbf{k}_a = (k_\ell : \ell \in N(a))$, and $k_\ell = [\hat{\mathbf{v}}(\ell)]_2$ with the function $\mathcal{G}_{a_p}$ defined in (9). Equivalently, k_ℓ can be interpreted as the index of the one in $\hat{\mathbf{m}}(\ell)$. In words, function $\mathcal{G}(\mathbf{k})$ verifies the parity consistency of its vector argument $\mathbf{k}$ under the parity structure defined in (8).

Having specified factor functions in (16), we can write standard expressions for the message passing rules. A message passed from check node a_p to variable node $s \in N(a_p)$ has the form

$$\boldsymbol{\mu}_{a_p \to s}(k) = \sum_{\mathbf{k}_{a_p} : k_p = k} \mathcal{G}_{a_p}(\mathbf{k}_{a_p}) \prod_{s_j \in N(a_p) \setminus s} \boldsymbol{\mu}_{s_j \to a_p}(k_j). \tag{17}$$

Similarly, a message going from variable node s_ℓ to check node $a \in N(s)$ abides by

$$\boldsymbol{\mu}_{s_\ell \to a}(k) \propto \boldsymbol{\lambda}_\ell(k) \prod_{a_p \in N(s_\ell)\setminus a} \boldsymbol{\mu}_{a_p \to s_\ell}(k). \tag{18}$$

The '$\propto$' symbol indicates that the measure is normalized before being sent out as a message. Block vector $\boldsymbol{\lambda}_\ell$ in (18) can be interpreted as a collection of local estimates, where entry $\boldsymbol{\lambda}_\ell(k)$ represents the estimate that a designated device has sent integer fragment k within section ℓ as part of its message. We have some flexibility in selecting a suitable estimator for each such component, but this value should be calculated solely based on the intrinsic information afforded by $\mathbf{r}(\ell)$, as is customary in the derivation of message passing rules. We delay the treatment of $\boldsymbol{\lambda}_\ell$ until Section IV. Still, it may be helpful to note that its components are closely linked to the likelihood ratio for binary classification in additive Gaussian noise, namely

$$\begin{aligned}\mathcal{L}_\ell(k) &= \exp\left(-\frac{(\mathbf{r}(\ell,k)-d_\ell)^2 - \mathbf{r}(\ell,k)^2}{2\tau^2}\right) \\ &= \exp\left(\frac{2d_\ell \mathbf{r}(\ell,k) - d_\ell^2}{2\tau^2}\right).\end{aligned} \tag{19}$$

Parameter τ in (19) corresponds to the standard deviation of the noise component in the effective observation of (15), where $\mathbf{r}(\ell) = d_\ell \mathbf{s}(\ell) + \tau\boldsymbol{\zeta}(\ell)$ and recall that d_ℓ is the amplitude of the signal for section ℓ. Following established notation, $\mathbf{r}(\ell)$ denotes the ℓth section of $\mathbf{r}$ and element $\mathbf{r}(\ell,k)$ refers to the kth entry of the ℓth section of $\mathbf{r}$. The message passing rules are depicted in Fig. 2. All the dynamic messages are initialized with $\boldsymbol{\mu}_{s\to a} = \mathbf{1}$ and $\boldsymbol{\mu}_{a\to s} = \mathbf{1}$. The parallel sum-product algorithm then iterates between (17) and (18).

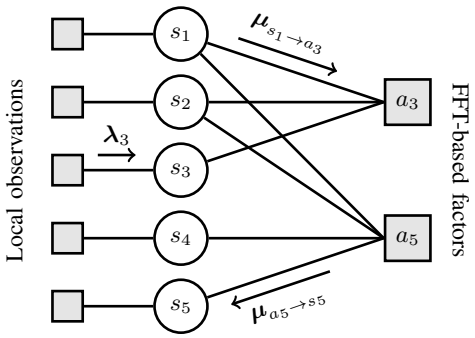

Fig. 2. This illustration shows the augmented factor graph for the outer code with variable nodes, parity check constraints, and extra factors associated with local observations. Local factor messages $\{\boldsymbol{\lambda}_\ell\}$ appear for mathematical convenience. They do not change during the belief propagation process when $\mathbf{r}$ is fixed.

At any stage of this iterative process, the belief vector on section ℓ based on extrinsic information is proportional to the product of the messages from adjoining parity factors. That is, it is proportional to

$$\boldsymbol{\mu}_{s_\ell}(k) = \prod_{a \in N(s_\ell)} \boldsymbol{\mu}_{a\to s_\ell}(k). \tag{20}$$

Likewise, the estimated marginal distribution of a specific device having transmitted index k at variable node s_ℓ is proportional to the product of the current messages from all adjoining factors, including the intrinsic information,

$$p_{s_\ell}(k) \propto \boldsymbol{\lambda}_\ell(k) \prod_{a \in N(s_\ell)} \boldsymbol{\mu}_{a\to s_\ell}(k) = \boldsymbol{\lambda}_\ell(k)\boldsymbol{\mu}_{s_\ell}(k). \tag{21}$$

That is, a normalized version of $\boldsymbol{\lambda}_\ell(k)\boldsymbol{\mu}_{s_\ell}(k)$ can be viewed as an estimate for the event $\{\mathbf{m}_i(\ell,k)=1\}$, where i is fixed. We refer the reader to Appendix B for a detailed explanation of this estimate and how it relates to belief propagation. The expected value of state vector $\mathbf{s}$ is given component-wise by

$$\begin{aligned}\hat{\mathbf{s}}(\ell,k) = \mathbb{E}\left[\mathbf{s}(\ell,k)|\mathbf{r}\right] &= \mathbb{E}\left[\sum_{i\in[K_\mathrm{a}]} \mathbf{m}_i(\ell,k)\middle|\mathbf{r}\right] \\ &= \sum_{i\in[K_\mathrm{a}]} \mathbb{E}\left[\mathbf{m}_i(\ell,k)|\mathbf{r}\right] \approx K_\mathrm{a} p_{s_\ell}(k).\end{aligned} \tag{22}$$

This is precisely the information needed to provide estimates for the support of $\mathbf{s}$ to the AMP denoiser. In general, such an iterative procedure is guaranteed to converge for acyclic graphical models, but not for arbitrary graphs [30]. Nevertheless, it is known to perform well in many cases where factor graphs have cycles. Having said that, it is intuitively appealing to construct outer codes whose factor graphs do not contain short cycles. In the framework we envision, the denoiser performs only one (or a select few) composite steps of the belief propagation algorithm before returning the updated state estimate $\hat{\mathbf{s}}$ back to the AMP algorithm, which subsequently seeks to improve the effective observation $\mathbf{r}$. We will come back to this particular point in Section IV, where the reason for halting belief propagation early can be explained adequately.

D. Design Considerations for Fast Execution

A naive implementation of (17) would have the denoiser parse through $\prod_{j:s_j\in N(a)} v_j$ distinct paths to compute message $\boldsymbol{\mu}_{a\to s}$, an approach which is intractable for the parameters of interest. Our goal, then, is to exploit the structure of the revised outer code to get a low-complexity solution. This is where the cyclic aspect of the parity precursors in (8) comes into play. Let $k = [\hat{\mathbf{v}}(\ell)]_2$ and define $g = [\hat{\mathbf{v}}(\ell)\mathbf{G}_{\ell,p}]_{\mathbb{Z}/2^{v_p}\mathbb{Z}}$. Then, we can write

$$\begin{aligned}\boldsymbol{\mu}_{a_p\to s_\ell}(k) &\propto \sum_{\mathbf{k}_{a_p}:k_p=k} \mathcal{G}_{a_p}\left(\mathbf{k}_{a_p}\right) \prod_{s_j\in N(a_p)\setminus s_\ell} \boldsymbol{\mu}_{s_j\to a_p}(k_j) \\ &= \underbrace{\sum_{\substack{\mathbf{g}_{a_p}: g_\ell = g \\ \sum g_j \equiv 0}} \prod_{s_j\in N(a_p)\setminus s_\ell} \left(\sum_{[\hat{\mathbf{v}}(j)\mathbf{G}_{j,p}]\equiv g_j} \boldsymbol{\mu}_{s_j\to a_p}\left([\hat{\mathbf{v}}(j)]_2\right)\right)}_{\text{circular convolution structure}},\end{aligned} \tag{23}$$

where we have implicitly introduced matrix $\mathbf{G}_{p,p} = -\mathbf{I}$ for the sake of exposition. This definition transforms parity check equation (8) into the symmetric form

$$\begin{aligned}0 &\equiv \sum_{j:s_j\in N(a_p)} [\mathbf{v}(j)\mathbf{G}_{j,p}]_{\mathbb{Z}/2^{v_p}\mathbb{Z}} \\ &\equiv \sum_{j:s_j\in N(a_p)} [\mathbf{v}(j)\mathbf{G}_{j,p}]_{\mathbb{Z}} \mod 2^{v_p}.\end{aligned}$$

The circular discrete convolution structure identified above suggests the application of the discrete Fourier transform and related techniques. In addition, since the underlying period is 2^{v_p} (a factor of two), these operations can be performed with the FFT algorithm. Thus, through the structure of the parity patterns produced by (8), the computation of messages becomes manageable under (23), even for large values of 2^{v_p}. This fact forms the impetus behind the development of the alternate outer code and the adoption of its more intricate parity generation process in Section III-A.

Pragmatically, message $\boldsymbol{\mu}_{a_p \to s_\ell}$, where $p \in \mathcal{P}$, can be computed as follows. For every j such that $s_j \in N(a_p)$, a collection of static binary vectors $\left\{\mathbf{g}_{j,p}^{(g)} \in \{0,1\}^{m_j}\right\}$ is maintained, one vector for every $g \in \mathbb{Z}/2^{v_p}\mathbb{Z}$. Vector $\mathbf{g}_{j,p}^{(g)}$ features a one at every index location where $\hat{\mathbf{v}}(j)$ is such that $[\hat{\mathbf{v}}(j)\mathbf{G}_{j,p}]_{\mathbb{Z}/2^{v_p}\mathbb{Z}} \equiv g$, and zeros everywhere else. Given factor $p \in \mathcal{P}$, this vector marks all the locations associated with parity precursor g at level j. Through this partitioning, the aggregate weight of this group becomes

$$\begin{aligned}\mathbf{L}_{j,p}(g) &= \sum_{[\hat{\mathbf{v}}(j)\mathbf{G}_{j.p}]_{\mathbb{Z}/2^{v_p}\mathbb{Z}} \equiv g} \boldsymbol{\mu}_{s_j \to a_p}\left([\hat{\mathbf{v}}(j)]_2\right) \\ &= \left\langle \boldsymbol{\mu}_{s_j \to a_p}, \mathbf{g}_{j,p}^{(g)} \right\rangle .\end{aligned} \tag{24}$$

When ordered and stacked, the values in (24) yield a vector $\mathbf{L}_{j,p} \in \mathbb{R}^{2^{v_p}}$. Given the circular convolution structure identified above, message $\boldsymbol{\mu}_{a_p \to s_\ell}$ can be computed at once as

$$\begin{aligned}&\boldsymbol{\mu}_{a_p \to s_\ell}\left([\hat{\mathbf{v}}(\ell)]_2\right) \\ &\propto \frac{1}{\left\|\mathbf{g}_{\ell,p}^{(g)}\right\|_0} \left(\mathrm{FFT}^{-1}\left(\prod_{s_j \in N(a_p)\setminus s_\ell} \mathrm{FFT}\left(\mathbf{L}_{j,p}\right) \right) \right)(g)\end{aligned} \tag{25}$$

where $[\hat{\mathbf{v}}(\ell)\mathbf{G}_{\ell,p}]_{\mathbb{Z}/2^{v_p}\mathbb{Z}} \equiv g$. The leading coefficient in (25) accounts for the fact that the weight of a certain parity precursor g within section ℓ, based on the information afforded by an adjoining factor, is evenly distributed among entries that map to g. This highlights the need to avoid situations where several entries of p_{s_ℓ} in (21) map to a same $g \in \mathbb{Z}/2^{v_p}\mathbb{Z}$. Again, the '$\propto$' symbol accounts for the normalization of the belief vector.

1) Decoding with Extended Lists: In the hard version of tree decoding, lists at various stages contain K_a entries (or slightly more), and paths moving forward are pruned aggressively. The tools described herein and, specifically, (25) enable the propagation of soft estimates over much larger collections. This is key in creating a soft denoising that accounts for the underlying outer code. Also, we see how having both information and parity bits within a same block is not conducive to the effective application of (25) because of weight splitting. Such a situation would reduce the cardinality of $\mathbb{Z}/2^{v_p}\mathbb{Z}$, thereby reducing the number of masks $\left\{\mathbf{g}_{j,p}^{(g)} : g \in \mathbb{Z}/2^{v_p}\mathbb{Z}\right\}$ and, coincidentally, lowering resolution. This justifies the approach we adopted in Section II-A: a codeword $\mathbf{v}$ contains homogeneous blocks of information bits, with sections of parity bits interspersed in-between. In view of these insights, we examine systems where sections of information bits are followed by a section of parity bits, as depicted in Fig 3.

Fig. 3. An information and parity allocation that is conducive to the application of FFT-based techniques appears above. Blocks are homogeneously composed of information or parity bits, but not both. The shaded component denotes a parity block.

2) Tree Pruning and Block Stitching: Our discussion so far has focused on how the outer code structure can be integrated into the composite AMP iteration. Once the AMP has converged and returned a reliable estimate for $\mathbf{s}$, the outer decoder must perform the stitching process that binds consistent blocks together. After stitching, the decoder outputs $\{\hat{\mathbf{m}}_1, \ldots, \hat{\mathbf{m}}_{K_\mathrm{a}}\}$ or, equivalently, $\widehat{W}(\mathbf{y})$, which concludes the decoding of $\mathbf{y}$. The stitching process for the original tree code is discussed at length in [4] and, as such, we do not reproduce this treatment in the present article. Still, it is pertinent to mention a couple implementation tricks that can be applied after AMP has converged to fuse blocks together. Note that, once AMP has converged and the iteration process is terminated, the effective observation remains fixed. One can keep iterating on the factor graph of the outer code, and progressively apply thresholding to the components of $(\boldsymbol{\lambda}_\ell : \ell \in [L])$, setting small entries to zero. Effectively, this turns very unlikely entries into impossible locations, which propagates over the factor graph through message passing in a natural manner.

A second interesting trick arises as a consequence of our redesigned homogeneous outer code: every block features information bits or parity bits, but not both. For such outer codes, binding can be performed on local neighborhoods of the form $(\mathbf{s}(j) : j \in N(a))$. Suppose that the lists in $N(a)$ have already been pruned with high statistical confidence as described above, leaving a few candidate blocks per section. Then, parity constraints can be enforced on aggregates of the surviving members (per section), resulting in a reduced list of most likely super-sections. This procedure is portrayed in Fig. 4, where 64 possible paths are distilled into two super-sections after halving individual lists and checking parity constraints. Extensions to local binding, beyond members of $N(a)$, are conceptually straightforward. For instance, repeated applications of these concepts can be scaffolded in a cascading, hierarchical, or mixed fashion.

E. Synopsis of Revised Outer Code

In summary, the revised outer code is designed to facilitate belief propagation on its factor graph at scale, while providing pertinent statistical information to the inner decoder at every step. This is accomplished by using homogeneous blocks composed of either information bits or parity bits, but not both. The parity bits are redefined to establish a circular convolution structure conducive to the application of FFT techniques. Yet, the revised design preserves the foundation on which the

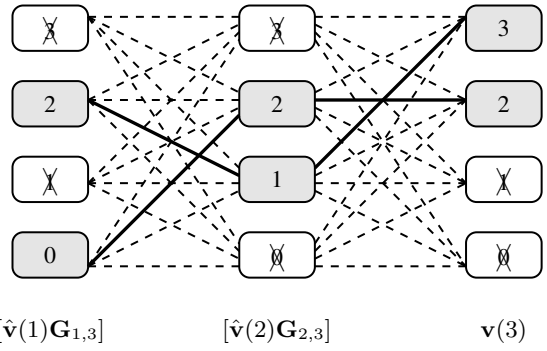

Fig. 4. In this example with $v_1 = v_2 = v_3 = 2$, the 64 parity consistent paths get pruned to two groupings (super-sections) after the belief vector of each list has concentrated on two elements.

analysis of [4] is built and, as such, performance guarantees can be obtained when the generator matrices are random binary matrices whose entries are independent Rademacher trials. Ultimately, performance also depends on the structure of the factor graph associated with the outer code.

IV. Inner Code and AMP Decoding

We are ready to initiate our description of the enhanced CCS-AMP algorithm. As mentioned in Section II, the inner code introduced in (7) operates on a received signal of the form

$$\mathbf{y} = \mathbf{A}\mathbf{D}\mathbf{s} + \mathbf{z} \tag{26}$$

where $\mathbf{s} = \mathbf{s}(1)\cdots\mathbf{s}(L)$ is fragmented into L sections. Recall that every section in $\mathbf{s}$ is $K_{\rm a}$-sparse because $\mathbf{s}(\ell) = \sum_{i=1}^{K_{\rm a}} \mathbf{m}_i(\ell)$, a structure which suggests that AMP can perform well in this setting. Matrix $\mathbf{A}$ is normalized in that the 2-norm of every column is one. Matrix $\mathbf{D}$ is diagonal with equal, non-negative diagonal entries within each section. It accounts for the transmit power allocated to each block; the amplitudes are labeled $d_\ell = \sqrt{nP_\ell}$. In the spirit of AMP for sparse regression codes [14], [15], [18], [20], the challenge in designing the AMP decoder is to create a composite iterative process to recover sparse vector $\mathbf{s}$. The composite algorithm iterates through two equations:

$$\mathbf{z}^{(t)} = \mathbf{y} - \mathbf{A}\mathbf{D}\mathbf{s}^{(t)} + \frac{\mathbf{z}^{(t-1)}}{n} \operatorname{div} \mathbf{D}\boldsymbol{\eta}_{t-1}\left(\mathbf{r}^{(t-1)}\right) \tag{27}$$

$$\mathbf{s}^{(t+1)} = \boldsymbol{\eta}_t\Big(\underbrace{\mathbf{A}^{\rm T}\mathbf{z}^{(t)} + \mathbf{D}\mathbf{s}^{(t)}}_{\mathbf{r}^{(t)}}\Big) \tag{28}$$

with initial conditions $\mathbf{s}^{(0)} = \mathbf{0}$ and $\mathbf{z}^{(0)} = \mathbf{y}$. The first equation can be interpreted as a computation of the *residual* enhanced with an Onsager correction [31], [32]. The second equation updates the state estimate through denoising. The collection of denoising functions $(\boldsymbol{\eta}_t(\cdot))_{t\geq 0}$ will be defined shortly. For the time being, it suffices to say that $\boldsymbol{\eta}_t(\cdot)$ seeks to leverage the structure embedded in $\mathbf{s}$ while computing a state update. The argument of the denoiser in (28), termed the *effective observation* $\mathbf{r}^{(t)}$, also plays an important role in the upcoming discussion. We emphasize that our specific AMP characterization falls within the extended framework for non-separable functions characterized by Berthier, Montanari, and Nguyen [33].

The first application of AMP to the unsourced MAC with a CCS tree outer code is due to Fengler, Jung, and Caire [5]. The authors therein demonstrate that AMP can be adapted to the application scenario at hand, although this is only possible after addressing several technical challenges rooted in the sparse structure of the problem. We briefly revisit some of these advances below for the sake of completeness. We also describe our contributions and point out places where our envisioned framework differs from the scheme put forth by Fengler et al.

A. Prior Art

To gain a better understanding of the proposed AMP decoder, we begin by looking at (28), where $\mathbf{r}^{(t)}$ acts as a test statistic. A remarkable fact about AMP is that, under certain conditions on the denoising functions, the effective observation $\mathbf{r}^{(t)}$ is asymptotically distributed as $\mathbf{D}\mathbf{s} + \tau_t\boldsymbol{\zeta}_t$ where $\boldsymbol{\zeta}_t$ is an i.i.d. $\mathcal{N}(0,1)$ random vector and τ_t is a deterministic quantity. This property typically hinges on the presence of an Onsager correction term in the iterative algorithm and it relies on the denoiser being sufficiently smooth [33]. In the original AMP for SPARCs implementation [15]–[17], which is designed for a single user, the denoiser is applied independently to every section. Specifically, the denoising function adopted therein is an instance of a minimum mean square error (MMSE) estimator that accounts for a one-sparse structure per block (single user). Given test statistic $\mathbf{r}(\ell)$, their updated block estimate takes the form

$$\begin{aligned}&\mathbb{E}\left[\mathbf{s}(\ell) | d_\ell\mathbf{s}(\ell) + \tau_t\boldsymbol{\zeta}_t(\ell) = \mathbf{r}(\ell)\right]\\ &\quad\propto \sum_{\mathbf{m}(\ell)} \mathbf{m}(\ell) \exp\left(-\frac{\|\mathbf{r}(\ell) - d_\ell\mathbf{m}(\ell)\|^2}{2\tau_t^2}\right)\end{aligned} \tag{29}$$

where '$\propto$' accounts for normalization and the sum is over the m_ℓ possible one-sparse blocks. While this approach works adequately for a per block sparsity of one and reasonably small sections, it does not extend easily to unsourced random access. In this latter case, computational complexity rapidly becomes overwhelming. Indeed, within a URA scenario, an optimal Bayesian denoiser must take into account the $K_{\rm a}$-sparse constraint on the support of $\mathbf{s}(\ell)$. That is, instead of the m_ℓ distinct possibilities in the original SPARC setting, an optimal solution must parse through $\binom{m_\ell}{K_{\rm a}}$ arrangements for the support of $\mathbf{s}(\ell)$, a computationally expensive task.

A suitable approximation to this solution, for select parameters, is the marginal posterior mean estimate (PME) of Fengler et al. [5]. This approximation relies on the assumption that collisions occur with a small probability and can safely be neglected. The marginal prior probability of the event $\{\mathbf{s}(\ell,k) = i\}$ is given by

$$\Pr(\mathbf{s}(\ell,k) = i) = \binom{K_{\rm a}}{i}\frac{1}{m_\ell^i}\left(1 - \frac{1}{m_\ell}\right)^{K_{\rm a}-i}.$$

Then, the probability that multiple active devices transmit index k during slot ℓ can be computed as

$$\begin{aligned}\sum_{i=2}^{K_{\rm a}} \Pr(\mathbf{s}(\ell,k)=i) &= 1-\Pr(\mathbf{s}(\ell,k)=0)-\Pr(\mathbf{s}(\ell,k)=1)\\ &= 1-\left(1-\frac{1}{m_\ell}\right)^{K_{\rm a}} - \frac{K_{\rm a}}{m_\ell}\left(1-\frac{1}{m_\ell}\right)^{K_{\rm a}-1}\\ &\approx \frac{K_{\rm a}(K_{\rm a}-1)}{m_\ell^2},\end{aligned}$$

which is negligible in the considered parameter regime since $K_{\rm a} \ll m_\ell$. To further explain the concept, we begin with an elementary building block. Consider scalar binary signal $s \in \{0,1\}$ embedded in Gaussian noise. Suppose the prior distribution for this signal is $q = \Pr(s=1) = 1-\Pr(s=0)$, and the observation model is $r = ds + \tau\zeta$. The PME, which is also the conditional probability that s is equal to one given observation r, is characterized in Lemma 3.

Lemma 3 (PME [5])**.** *The posterior mean estimator (PME) for binary signal s, conditioned on observation $r = d_\ell s + \tau\zeta$ where $\zeta \sim \mathcal{N}(0,1)$, takes the form*

$$\hat{s}_\ell(q,r,\tau) = \frac{q\exp\left(-\frac{(r-d_\ell)^2}{2\tau^2}\right)}{q\exp\left(-\frac{(r-d_\ell)^2}{2\tau^2}\right)+(1-q)\exp\left(-\frac{r^2}{2\tau^2}\right)}. \tag{30}$$

where q is the prior probability of entry s being equal to one, and $(1-q)$ is the probability of it being zero.

Proof: For a random variable s taking value one with probability q and zero otherwise, the conditional expectation can be written as

$$\hat{s}_\ell(q,r,\tau) = \mathbb{E}\left[s | d_\ell s + \tau\zeta = r\right],$$

where ζ is a standard Gaussian random variable independent of s. Therefore, we get the explicit formula

$$\begin{aligned}\hat{s}_\ell(q,r,\tau) &= \frac{0\cdot(1-q)f\left(\frac{r}{\tau}\right)+1\cdot qf\left(\frac{r-d_\ell}{\tau}\right)}{(1-q)f\left(\frac{r}{\tau}\right)+qf\left(\frac{r-d_\ell}{\tau}\right)}\\ &= \frac{q\exp\left(-\frac{(r-d_\ell)^2}{2\tau^2}\right)}{(1-q)\exp\left(-\frac{r^2}{2\tau^2}\right)+q\exp\left(-\frac{(r-d_\ell)^2}{2\tau^2}\right)},\end{aligned}$$

where $f(\cdot)$ is the probability density function of a normal random variable. ∎

Constant $d_\ell = \sqrt{nP_\ell}$ in (30) comes from the value along the diagonal of $\mathbf{D}$ in the ℓth section; it captures the amplitude of the transmitted symbol. The purpose of this rudimentary lemma is to lay a foundation for the upcoming denoising functions. Another quantity that we will need shortly is the partial derivative of the marginal PME found in Lemma 4. It is instructive to highlight the close resemblance between $\hat{s}_\ell(q,r,\tau)$ and the logistic function before stating the lemma. In view of this connection, the form of the derivative should not come as a surprise.

Lemma 4. *The partial derivative with respect to r of the posterior mean estimator (PME) defined in Lemma 3 is*

$$\frac{\partial\hat{s}_\ell(q,r,\tau)}{\partial r} = \frac{d_\ell}{\tau^2}\hat{s}_\ell(q,r,\tau)\left(1-\hat{s}_\ell(q,r,\tau)\right). \tag{31}$$

Proof: First, we note that the PME can be rewritten as

$$\hat{s}_\ell(q,r,\tau) = \frac{q}{q+(1-q)\exp\left(\frac{d_\ell^2-2rd_\ell}{2\tau^2}\right)}. \tag{32}$$

Then, by the chain rule of differentiation, we get

$$\begin{aligned}\frac{\partial\hat{s}_\ell(q,r,\tau)}{\partial r} &= \frac{d_\ell}{\tau^2}\frac{q(1-q)\exp\left(\frac{d_\ell^2-2rd_\ell}{2\tau^2}\right)}{\left(q+(1-q)\exp\left(\frac{d_\ell^2-2rd_\ell}{2\tau^2}\right)\right)^2}\\ &= \frac{d_\ell}{\tau^2}\hat{s}_\ell(q,r,\tau)\left(1-\hat{s}_\ell(q,r,\tau)\right),\end{aligned}$$

as stated. ∎

The denoiser introduced by Fengler et al. [5] can be described using the PME of Lemma 4 as follows. First, recall that q act as a proxy for the probability that the signal is non-zero. If there are m_ℓ locations in $\mathbf{s}(\ell)$ and $K_{\rm a}$ active devices, each picking a location independently, then the probability that any particular entry is non-zero can be expressed as $q_\ell = 1-\left(1-\frac{1}{m_\ell}\right)^{K_{\rm a}} \approx \frac{K_{\rm a}}{m_\ell}$. Estimates for the entries in $\mathbf{s}(\ell)$, given observation $\mathbf{r}(\ell)$, can be obtained via

$$\hat{\mathbf{s}}_\ell^{\rm OR}\left(\mathbf{r}(\ell),\tau\right) = \left(\hat{s}_\ell\left(q_\ell,\mathbf{r}(\ell,k),\tau\right) : k \in 0,\ldots,m_\ell-1\right) \tag{33}$$

where $\mathbf{r}(\ell,k)$ denotes the kth entry of the ℓth section of $\mathbf{r}$, and q_ℓ is the fixed constant mentioned above. Their overall vector estimate is obtained by concatenating L blocks,

$$\hat{\mathbf{s}}^{\rm OR}\left(\mathbf{r},\tau\right) = \hat{\mathbf{s}}_1^{\rm OR}\left(\mathbf{r}(1),\tau\right)\cdots\hat{\mathbf{s}}_L^{\rm OR}\left(\mathbf{r}(L),\tau\right). \tag{34}$$

This low-complexity strategy offers very good empirical performance when combined with AMP, as reported in [5]. When integrated within the AMP composite algorithm, (27)–(28), the denoiser defined in (34) yields section estimate

$$\hat{\mathbf{s}}_\ell^{\rm OR}\left(\mathbf{A}^{\rm T}\mathbf{z}^{(t)}(\ell)+\mathbf{D}\mathbf{s}^{(t)}(\ell),\tau_t\right). \tag{35}$$

The state vector is then updated according to (35) utilizing the aggregate form of (34). This is a well-behaved denoiser, with desirable properties.

Lemma 5. *The denoiser $\hat{\mathbf{s}}^{\rm OR}\left(\mathbf{r},\tau\right)$ is Lipschitz continuous.*

Proof: From Lemma 4 and the fact that $\hat{s}_\ell(q,r,\tau) \in [0,1]$, we gather that

$$\left|\frac{\partial\hat{s}_\ell(q,r,\tau)}{\partial r}\right| \le \frac{d_\ell}{\tau^2}. \tag{36}$$

Moreover, we have $\frac{\partial\hat{s}_\ell(q_\ell,\mathbf{r}(\ell,k_\ell),\tau)}{\partial\mathbf{r}(j,k)} = 0$ whenever $j \neq \ell$ or $k \neq k_\ell$. It follows that, by the mean value theorem,

$$\begin{aligned}&\left\|\hat{\mathbf{s}}^{\rm OR}\left(\mathbf{r},\tau\right)-\hat{\mathbf{s}}^{\rm OR}\left(\mathbf{r}',\tau\right)\right\|^2\\ &= \sum_{\ell\in[L]}\sum_{k_\ell=0}^{m_\ell-1}\left(\hat{s}_\ell\left(q_\ell,\mathbf{r}(\ell,k_\ell),\tau\right)-\hat{s}_\ell\left(q_\ell,\mathbf{r}'(\ell,k_\ell),\tau\right)\right)^2\\ &\le \sum_{\ell\in[L]}\sum_{k_\ell=0}^{m_\ell-1}\left(\frac{d_\ell}{\tau^2}\right)^2\left(\mathbf{r}(\ell,k_\ell)-\mathbf{r}'(\ell,k_\ell)\right)^2\\ &\le \left(\frac{d_{\max}}{\tau^2}\right)^2\left\|\mathbf{r}-\mathbf{r}'\right\|^2\end{aligned}$$

where $d_{\max} = \max_\ell d_\ell$. This establishes the Lipschitz continuity of this denoiser. ∎

A naive explanation for this denoiser is that it seeks to compute

$$\mathbb{E}\left[\mathbf{s}(\ell,k)\middle|\sqrt{nP_\ell}\mathbf{s}(\ell,k)+\tau_t\boldsymbol{\zeta}_t(\ell,k)=\mathbf{r}^{(t)}(\ell,k)\right], \quad (37)$$

This is a loose interpretation because $\hat{s}_\ell\left(q_\ell,\mathbf{r}(\ell,k),\tau\right)$ disregards the fact that, with low probability, $\mathbf{s}(\ell,k)$ can take integer values other than zero or one. Nonethless, it is very close to (29) in spirit. A key insight behind this approach is that, instead of pursuing the computationally challenging task of estimating signal $\mathbf{s}$ optimally from the effective observation, it suffices to infer its support using marginal distributions.

B. Dynamic PME Denoising

The alternate viewpoint we propose in this article stems from the realization that, given the presence of an outer code, estimates of the elements of $\mathbf{s}(\ell)$ can be improved by taking advantage of the underlying code construction, as described in Section III. Adopting constant q_ℓ in (33) disregards the factor graph structure of the outer code altogether. It is equivalent to using uninformative prior marginal distributions within the PME at every stage. This situation reveals an opportunity to enhance CCS-AMP by accounting for connections between neighboring blocks through local factors. This alternative approach forms a marked departure from AMP for SPARCs in the context of the unsourced MAC [5]. It has the potential to accelerate convergence and improve performance significantly. An incentive to explore this research direction is that, although $\mathbf{v}$ is a vector of length 2^v, the outer encoding process essentially forces it to lie in a much smaller space that contains only 2^w elements. Likewise, the outer code confines every section of $\mathbf{s}$ to take value in a potentially much smaller subset when conditioned on graph neighboring sections through local factors. The factor graph structure of the outer code, paired with message passing, gives rise to beliefs for the components of $\mathbf{s}$ based on extrinsic information, as highlighted in (20). These can play a role equivalent to prior probability vector $\mathbf{q}(\ell)$ in evaluating (37). Altogether, exploiting the graphical structure of the revised outer code is attainable via FFT techniques, and it can help improve the inference process. Thus, our next goal is to define a denoiser in a manner analogous to (33), but to incorporate the extrinsic information embedded in $\left\{\mathbf{r}^{(t)}(j): j\in[L]\setminus\ell\right\}$ (or a good approximation thereof) in calculating estimates for the elements of $\mathbf{s}^{(t+1)}(\ell)$.

Formally, we propose to create denoiser function $\boldsymbol{\eta}_t^{\mathrm{PME}}\left(\mathbf{r}^{(t)}\right)$ as follows. We initiate local estimate vector $\boldsymbol{\lambda}_j^{\mathrm{PME}}(k)$ using (33), with

$$\boldsymbol{\lambda}_j^{\mathrm{PME}}(k)=\hat{s}_j\left(q_j,\mathbf{r}(j,k),\tau_t\right) \quad (38)$$

and $q_j = 1-(1-1/m_j)^{K_\mathrm{a}} \approx K_\mathrm{a}/m_j$. We then run a few rounds of belief propagation using message passing rules according to (17) and (18). We aggregate the messages coming to variable node s_ℓ from adjoining parity factors and compute belief vector

$$\mathbf{q}(\ell,k)=1-\left(1-\frac{\boldsymbol{\mu}_{s_\ell}(k)}{\|\boldsymbol{\mu}_{s_\ell}\|_1}\right)^{K_\mathrm{a}} \quad (39)$$

based on extrinsic information by applying (39). As a final denoising step, we compute section estimate using (30) in Lemma 3. The culmination of these steps leads to the following denoiser.

Definition 6 (Dynamic PME Denoiser)**.** *The functions in* (28) *for the dynamic PME denoiser are given by*

$$\boldsymbol{\eta}_t^{\mathrm{PME}}(\mathbf{r})=\hat{\mathbf{s}}_1^{\mathrm{PME}}\left(\mathbf{r},\tau_t\right)\cdots\hat{\mathbf{s}}_L^{\mathrm{PME}}\left(\mathbf{r},\tau_t\right). \quad (40)$$

where individual components are equal to

$$\begin{aligned}\hat{\mathbf{s}}_\ell^{\mathrm{PME}}\left(\mathbf{r},\tau_t\right)&=\left(\hat{s}_\ell^{\mathrm{PME}}\left(k,\mathbf{r},\tau_t\right):k\in0,\ldots,m_\ell-1\right)\\&=\left(\hat{s}_\ell\left(\mathbf{q}(\ell,k),\mathbf{r}(\ell,k),\tau_t\right):k\in0,\ldots,m_\ell-1\right).\end{aligned} \quad (41)$$

The effective observation $\mathbf{r}=\mathbf{A}^{\mathrm{T}}\mathbf{z}+\mathbf{D}\mathbf{s}$ *is provided by the AMP algorithm, whereas the belief vector* $\mathbf{q}(\ell)$ *is derived from extrinsic information using* (20) *and* (39) *with initial conditions* $\boldsymbol{\lambda}_j^{\mathrm{PME}}$*. Constants* τ_t^2 *in* (41) *can be obtained deterministically through the state evolution, which we discuss later. Alternatively, they can be approximated as* $\tau_t^2\approx\left\|\mathbf{z}^{(t)}\right\|^2/n$ *for* $t\geq0$ *[34].*

It is worth mentioning that the denoiser in Definition 6 reduces to the PME with uninformative prior probabilities introduced in [5] when the number of composite steps performed on the factor graph of the outer code is zero. Next, we turn to the divergence of $\mathbf{D}\boldsymbol{\eta}_t^{\mathrm{PME}}\left(\mathbf{r}\right)$, which gives rise to the Onsager correction term found in (27). Although $\boldsymbol{\eta}_t^{\mathrm{PME}}(\cdot)$ is a non-separable function, its divergence admits an elegant and tractable structure, provided that $\mathbf{q}(\ell)$ is computed based exclusively on extrinsic information. This condition can be ensured by requiring that the number of BP iterations on the factor graph be strictly less than its shortest cycle. In fact, this is the main motivation behind introducing this constraint on the number of BP iterations. When this condition applies, $\mathbf{q}(\ell)$ depends exclusively on $\{\mathbf{r}(j):j\in[L]\setminus\ell\}$; we will soon see how this is also highly desirable for the computation of the Onsager correction. We begin with a preliminary result that focuses on individual components.

Lemma 7. *If the number of message passing iterations on the factor graph is strictly less than the length of its shortest cycle, then the partial derivative of* $\hat{s}_\ell^{\mathrm{PME}}\left(k,\mathbf{r},\tau\right)$ *with respect to* $\mathbf{r}\left(\ell,k\right)$ *is given by*

$$\frac{\partial\hat{s}_\ell^{\mathrm{PME}}\left(k,\mathbf{r},\tau\right)}{\partial\mathbf{r}\left(\ell,k\right)}=\frac{d_\ell}{\tau^2}\hat{s}_\ell^{\mathrm{PME}}\left(k,\mathbf{r},\tau\right)\left(1-\hat{s}_\ell^{\mathrm{PME}}\left(k,\mathbf{r},\tau\right)\right). \quad (42)$$

Proof: We note that, predicated on the number of BP iterations being strictly less than the shortest loop in the factor graph, we are guaranteed to have

$$\frac{\partial\mathbf{q}(\ell,k)}{\partial\mathbf{r}\left(\ell,k\right)}=0$$

because $\mathbf{q}(\ell)$ only depends on extrinsic information or, equivalently, it is computed based on $\{\mathbf{r}(j):j\in[L]\setminus\ell\}$. Also,

recall that τ is a deterministic constant. Consequently, using the chain rule of differentiation, we get

$$\frac{\partial \hat{s}_\ell^{\mathrm{PME}}(k,\mathbf{r},\tau)}{\partial \mathbf{r}(\ell,k)} = \frac{\partial \hat{s}_\ell(\mathbf{q}(\ell,k),\mathbf{r}(\ell,k),\tau)}{\partial \mathbf{r}(\ell,k)}$$
$$= \frac{d_\ell}{\tau^2}\hat{s}_\ell(\mathbf{q}(\ell,k),\mathbf{r}(\ell,k),\tau)\left(1-\hat{s}_\ell(\mathbf{q}(\ell,k),\mathbf{r}(\ell,k),\tau)\right)$$
$$= \frac{d_\ell}{\tau^2}\hat{s}_\ell^{\mathrm{PME}}(k,\mathbf{r},\tau)\left(1-\hat{s}_\ell^{\mathrm{PME}}(k,\mathbf{r},\tau)\right)$$

The partial derivative of $\hat{s}_\ell(q,r,\tau)$ with respect to r comes from Lemma 4. ∎

Interestingly, the derivative in (42) does not depend on the number of BP iterations computed on the factor graph, provided that the conditions of Lemma 7 are met. The divergence associated with the denoiser is obtained below.

Proposition 8. *The divergence of* $\mathbf{D}\boldsymbol{\eta}_t^{\mathrm{PME}}(\mathbf{r})$ *with respect to* $\mathbf{r}$ *is equal to*

$$\operatorname{div}\mathbf{D}\boldsymbol{\eta}_t^{\mathrm{PME}}(\mathbf{r}) = \frac{1}{\tau_t^2}\left(\left\|\mathbf{D}^2\boldsymbol{\eta}_t^{\mathrm{PME}}(\mathbf{r})\right\|_1 - \left\|\mathbf{D}\boldsymbol{\eta}_t^{\mathrm{PME}}(\mathbf{r})\right\|^2\right). \tag{43}$$

Proof: First, we expand the div operator as

$$\operatorname{div}\mathbf{D}\boldsymbol{\eta}_t^{\mathrm{PME}}(\mathbf{r}) = \sum_{\ell=1}^{L} d_\ell \operatorname{div}\hat{\mathbf{s}}_\ell^{\mathrm{PME}}(\mathbf{r},\tau_t)$$
$$= \sum_{\ell=1}^{L} d_\ell \sum_{k=0}^{m_\ell-1} \frac{\partial \hat{s}_\ell^{\mathrm{PME}}(k,\mathbf{r},\tau_t)}{\partial \mathbf{r}(\ell,k)}$$
$$= \sum_{\ell=1}^{L} d_\ell \sum_{k=0}^{m_\ell-1} \frac{d_\ell}{\tau_t^2}\hat{s}_\ell^{\mathrm{PME}}(k,\mathbf{r},\tau_t)\left(1-\hat{s}_\ell^{\mathrm{PME}}(k,\mathbf{r},\tau_t)\right). \tag{44}$$

The last equality follows from substituting the expression for the partial derivative of $\hat{s}_\ell^{\mathrm{PME}}(k,\mathbf{r},\tau_t)$ obtained in Lemma 7. Again, we emphasize that $\mathbf{q}(\ell,k)$, being derived from extrinsic information, does not depend on the elements of block $\mathbf{r}(\ell)$, which implies $\frac{\partial \mathbf{q}(\ell,k)}{\partial \mathbf{r}(\ell,k)} = 0$. This property greatly limits the difficulty in computing the summands of (44). Recognizing $\left\{\hat{s}_\ell^{\mathrm{PME}}(k,\mathbf{r},\tau_t)\right\}$ as the components of $\boldsymbol{\eta}_t^{\mathrm{PME}}(\mathbf{r})$, the equation above can be interpreted as an inner product. Accounting for the signal amplitudes through diagonal matrix $\mathbf{D}$ yields

$$\operatorname{div}\mathbf{D}\boldsymbol{\eta}_t^{\mathrm{PME}}(\mathbf{r}) = \frac{1}{\tau_t^2}\left\langle \mathbf{D}\boldsymbol{\eta}_t^{\mathrm{PME}}(\mathbf{r}), \mathbf{D}\mathbf{1} - \mathbf{D}\boldsymbol{\eta}_t^{\mathrm{PME}}(\mathbf{r})\right\rangle$$
$$= \frac{1}{\tau_t^2}\left(\left\|\mathbf{D}^2\boldsymbol{\eta}_t^{\mathrm{PME}}(\mathbf{r})\right\|_1 - \left\|\mathbf{D}\boldsymbol{\eta}_t^{\mathrm{PME}}(\mathbf{r})\right\|^2\right) \tag{45}$$

where $\mathbf{1}$ is a vector whose entries are all ones. This is precisely the format of the Onsager correction in (43). ∎

Proposition 8, together with the facts that $\mathbf{A}$ is a normalized matrix and $\mathbf{s}^{(t)} = \boldsymbol{\eta}_{t-1}^{\mathrm{PME}}\left(\mathbf{r}^{(t-1)}\right)$, yields correction coefficient

$$\frac{1}{n}\operatorname{div}\mathbf{D}\boldsymbol{\eta}_{t-1}^{\mathrm{PME}}\left(\mathbf{r}^{(t-1)}\right) = \frac{1}{n\tau_{t-1}^2}\left(\left\|\mathbf{D}^2\mathbf{s}^{(t)}\right\|_1 - \left\|\mathbf{D}\mathbf{s}^{(t)}\right\|^2\right). \tag{46}$$

Overall, this produces an AMP algorithm that takes advantage of the structure of the underlying outer code throughout the decoding of the inner code. In our implementation, we update $\mathbf{q}(\ell)$ dynamically at every AMP iteration, essentially by running BP on a truncated factor graph. Still, it is possible to estimate $\operatorname{div}\mathbf{D}\boldsymbol{\eta}_t^{\mathrm{PME}}(\mathbf{r})$ in practice, without updating $\mathbf{q}(\ell)$ at every iteration, although such a variant would deviate from a strict AMP definition. The dynamical PME denoising is an appealing solution because it meshes nicely with prior art. It offers a conceptual bridge between using uninformative prior probabilities and performing multiple rounds of message passing on the factor graph of the outer code.

The dynamic PME denoiser is not as smooth as the original PME denoiser. This can be seen through the fact that collections of impossible entries in neighboring sections may lead to vanishing elements in $\mathbf{q}(\ell)$. From a decoding perspective, this is desirable because it can rapidly prune down the space of possibilities. However, it is much more difficult to obtain Lipschitz conditions under such circumstances. In Appendix C, we show that the dynamic PME denoiser with one round of message passing on the factor graph of the outer code is Lipschitz continuous. This property brings credibility to our upcoming study of the state evolution.

C. Good Outer Code Structures

The state evolution for CCS-AMP is intimately linked to the structure of the outer code whenever the algorithm dynamically updates the state estimate as part of every AMP composite iteration. To boost the benefits of incorporating the outer code within the AMP framework, a careful redesign of its graph structure is necessary. The parity allocation of the original tree code aims at limiting the growth of active paths during sequential tree decoding, while also maintaining high performance [4]. This leads to a parity assignment where information bits and parity bits coexist in several blocks. Yet, as mentioned earlier, such a strategy is not conducive to fast transform methods whereby all possible paths are assessed concurrently. Rather, it becomes desirable within CCS-AMP to have homogeneous blocks composed of either information bits or parity bits, but not both. Naively, the discriminating power of block ℓ is 2^{-p_ℓ}, and it is most effective when $p_\ell = v_\ell$. Moreover, heterogeneous blocks introduce dependencies that complicate the straightforward implementation of message passing with fast transform methods. This leads us to restrict our attention to outer codes with homogeneous blocks.

Another guiding principle for the redesign of the outer code stems from considerations associated with the mixing that takes place when multiple codewords are present on the same graph. Indeed, parity precursors coming from different codewords are interpreted as producing likely parity patterns when message passing is applied to the outer code. In particular, when sections of parity bits are interspersed in-between information blocks, the number of likely patterns associated with a parity section is equal to the product of the number of likely parity precursors in every precursory section. Thus, for a parity section to remain informative while running AMP iterations, it should not be created by combining too many information sections. We elucidate this phenomenon through a crude, motivating example below.

Example 9. *Consider a rudimentary outer code with three sections, $\ell = 1, 2, 3$, as in Fig. 3. Suppose that all the sections share a common size of $m_\ell = 2^{16}$ and that they are homogeneous, containing either information bits or parity bits, but not both. Let K_a be the number of active devices. Also, for illustrative purposes, suppose that the probability distribution associated with each section has concentrated over H entries of $\hat{\mathbf{s}}(\ell)$, including the K_a legitimate message indices. We wish to estimate the expected number of indices in each section that retain a high probability after the parity constraints have been enforced. We note that, by design, all the legitimate indices are validated and maintained. Consider an erroneous parity pattern, i.e., one that does not arise from genuine parity precursors in previous sections. For this pattern to survive the validation step, it must pair up with probable indices in the preceding sections in a parity consistent manner. When parities are created using two information sections, there are (at most) H^2 admissible parity consistent patterns based on likely neighbors, out of which $H^2 - K_\mathrm{a}$ are erroneous. The probability that a randomly selected, erroneous pattern in section 3 falls within the set of erroneous parity consistent patterns is approximately*

$$\frac{H^2 - K_\mathrm{a}}{2^{16} - K_\mathrm{a}} \quad \left(H < 2^8\right).$$

Given that there are $H - K_\mathrm{a}$ likely erroneous parity patterns before these constraints are checked, the expected number of surviving erroneous indices after applying the local factor inherited from the outer code becomes approximately

$$\frac{(H - K_\mathrm{a})(H^2 - K_\mathrm{a})}{2^{16} - K_\mathrm{a}} \quad \left(H < 2^8\right).$$

Consequently, as distributions start to concentrate over $H = \sqrt{2^{16}} = 256$ indices, the parity constraints of the outer code begin to softly discount erroneous blocks. This accelerates the AMP convergence process and reduces the probability of erroneous indices surviving the decoding of the inner code. In comparison, when the parity patterns act on three information sections, a similar pruning effect only starts to take place once the distributions have concentrated on approximately $H = \sqrt[3]{2^{16}} \approx 40$ values. That is, the probability that an erroneous block is parity consistent with other sections remains high for much longer. While it is possible to calculate $\left\{\boldsymbol{\mu}_{s_\ell}\right\}$ in such cases, the statistics obtained remain uninformative for many more AMP cycles, which nullifies the potential benefits of the enhanced CCS-AMP algorithm with its dynamic denoiser. This becomes progressively worse as the number of information sections over which parity patterns are computed increases.

A third consideration pertaining to the design of our revised outer code is the fact that we should avoid short cycles in the factor graph representation. Empirically, factor graphs that only contain long cycles are amenable to belief propagation, whereas message passing over factor graphs with short loops may be problematic. Of course, the fact that Lemma 7 only applies when the number of message passing iterations on the factor graph is strictly less than the length of its shortest cycle is also an incentive to design outer codes with no short cycles.

Given these observations, combined with the evidence afforded by numerical simulations, we focus mostly on outer structures based on triadic designs whereby parity sections are determined based on two of the preceding blocks. Although Example 9 overlooks issues such as correlations and feedback, the results obtained by applying these guidelines are excellent for the operating parameters we are interested in. Moreover, alternate graphical constructions for the outer code where parity sections are formed using more than two precursory sections do not perform as well empirically. For these reasons, our analysis of CCS-AMP moving forward assumes a triadic structure such as the construction depicted in Fig. 5. That is, every parity section is attached to two other blocks, and short loops are avoided altogether in the factor graph.

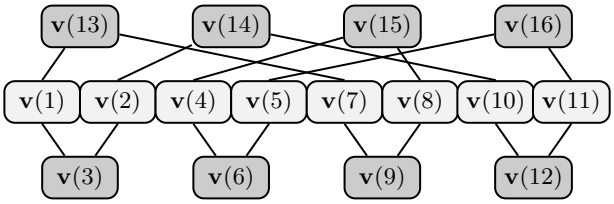

Fig. 5. This graph shows the type of triadic connections between precursory information sections and parity blocks utilized in the design of the revised outer code. Shaded blocks denote parity sections, whereas light blocks correspond to information fragments.

We point out, again, that the proposed approach warrants making a decision about the structure of the outer code before being able to tackle the state evolution in AMP. This differs from the algorithm presented in [5], whereby the state evolution is completely detached from the outer code. With a candidate design class in mind, we proceed to evaluating the progression of the AMP algorithm.

D. State Evolution

The state evolution is a standard object in the treatment of AMP [31]. It offers a blueprint to determine the sequence $\{\tau_t^2\}_{t \geq 0}$ and, concurrently, it provides a predictor of algorithmic performance. The computations for the state evolution rest on the fact that, asymptotically in the number of dimensions, the effective observations become Gaussian. Specifically, the analysis hinges on $\mathbf{r}^{(t)}$ being distributed according to

$$\mathbf{r}^{(t)} \sim \mathbf{D}\mathbf{s} + \tau_t \boldsymbol{\zeta}_t \tag{47}$$

where $\boldsymbol{\zeta}_t$ is an i.i.d. $\mathcal{N}(0, 1)$ random vector, as mentioned at the beginning of Section IV-A. This broad setting is common to most AMP analyses. The standard deviation parameter τ_t is computed iteratively.

We delayed the treatment of the state evolution until now because it is predicated on the structure of the outer code for the proposed CCS-AMP framework. Indeed, this is a byproduct of the fact that the outer code appears within the denoising functions and, consequently, its structure must be specified explicitly. Likewise, defining the outer code is needed to examine the performance benefit associated with our more intricate scheme when compared to the performance of the original, tree-agnostic version in [5]. Below, we study designs

wherein every information section is involved in exactly two disjoint triadic parity connections. The rationale for this condition can be found in Section IV-C. This structure is general enough to accommodate the scenarios we are interested in, and to provide insight on the suitability of competing designs. Finally, we assume that there are no collisions at the section level; that is, $\|\mathbf{s}(\ell)\|_0 = K_{\rm a}$ for $\ell \in [L]$. This is typical for sparse settings with a finite number of levels.

The state evolution captures the progression of the AMP algorithm, as a function of the iteration count t. We parallel the development found in [33] to characterize the evolution of this system, and we specialize their results to the application at hand. Define

$$\begin{aligned}\hat{\mathbf{z}}^{(t)} &= \sigma_t \boldsymbol{\xi}_t \\ \hat{\mathbf{r}}^{(t)} &= \mathbf{D}\mathbf{s} + \tau_t \boldsymbol{\zeta}_t\end{aligned}$$

where $\boldsymbol{\xi}_t$ and $\boldsymbol{\zeta}_t$ are i.i.d. $\mathcal{N}(0,1)$ random vectors. The values of parameters σ_t and τ_t can be obtained through a composite iteration process. The first equation for the variance parameters is simply

$$\begin{aligned}\tau_t^2 &= \sigma^2 + \lim_{n\to\infty} \frac{1}{n}\mathbb{E}\left[\left\langle \hat{\mathbf{z}}^{(t)}, \hat{\mathbf{z}}^{(t)} \right\rangle\right] \\ &= \sigma^2 + \lim_{n\to\infty} \frac{1}{n}\mathbb{E}\left[\left\|\sigma_t \boldsymbol{\xi}_t\right\|^2\right] = \sigma^2 + \sigma_t^2,\end{aligned} \tag{48}$$

where σ^2 is the variance of the observation noise in (1). The second equation is more contrived with

$$\sigma_{t+1}^2 = \lim_{n\to\infty} \frac{1}{n}\mathbb{E}\left[\left\|\mathbf{D}\left(\boldsymbol{\eta}_t\left(\mathbf{D}\mathbf{s} + \tau_t\boldsymbol{\zeta}_t\right) - \mathbf{s}\right)\right\|^2\right]. \tag{49}$$

The initial conditions are $\tau_0^2 = \sigma^2 + \sigma_0^2 = \lim_{n\to\infty}\|\mathbf{y}\|^2/n$. The foundation of the state evolution is rooted in the following proposition.

Proposition 10 (Berthier, Montanari, and Nguyen)**.** *Let the AMP iteration* $\left\{\mathbf{z}^{(t)}, \mathbf{r}^{(t)}\right\}_{t\geq 1}$ *be generated via* (27) *and* (28) *with initial conditions* $\mathbf{z}^{(0)} = \mathbf{y}$*,* $\mathbf{s}^{(0)} = \mathbf{0}$ *and assuming that* τ_t *is taken from* (48)*. Consider the state evolution* $\left\{\hat{\mathbf{z}}^{(t)}, \hat{\mathbf{r}}^{(t)}\right\}_{t\geq 1}$ *where the variance parameters are also those defined in* (48) *with initial condition* $\tau_0 = \lim_{n\to\infty}\|\mathbf{y}\|/\sqrt{n}$*. Then, under some regularity conditions,* $\mathbf{z}^{(t)} \to \hat{\mathbf{z}}^{(t)}$ *and* $\mathbf{r}^{(t)} \to \hat{\mathbf{r}}^{(t)}$ *in probability for all* $t \geq 0$*.*

Proof: This proposition is a restriction of Theorem 1 and Corollary 2 in [33]. In applying this result, it is pertinent to mention that its regularity conditions are fulfilled. In particular, $\mathbf{A}$ is an i.i.d. Gaussian matrix with normalized columns. Also, from Lemma 5, the original PME denoiser is (uniformly) Lipschitz. For the triadic designs considered in this paper, we show in Appendix C that the dynamic PME denoiser is (uniformly) Lipschitz when one composite BP step is performed on the outer factor graph per AMP iteration. This implies that the state evolution is also accurate in the presence of this dynamic PME denoiser. ∎

In order to leverage the state evolution to compare the two denoisers discussed thus far, we must compute (or approximate) the right-hand-side of (49) for these various cases. We begin by writing

$$\begin{aligned}&\mathbb{E}\left[\left\|\mathbf{D}\left(\boldsymbol{\eta}_t\left(\mathbf{D}\mathbf{s} + \tau_t\boldsymbol{\zeta}_t\right) - \mathbf{s}\right)\right\|^2\right] \\ &= \sum_{\ell\in[L]} d_\ell^2 \mathbb{E}\left[\left\|\hat{\mathbf{s}}_\ell\left(d_\ell \mathbf{s}(\ell) + \tau_t\boldsymbol{\zeta}_t(\ell), \tau_t\right) - \mathbf{s}(\ell)\right\|^2\right].\end{aligned} \tag{50}$$

We note that this equation has the same form irrespective of the number of BP iterations in the denoiser. The challenge in applying (50) comes from the fact that a closed-form expression is not available and a numerical evaluation of this expectation involves a very large number of random components.

a) Original PME Denoiser: We explore the specifics of the state evolution by first considering the simpler case: the original posterior mean estimate without message passing. There are exactly $K_{\rm a}$ locations in vector $\mathbf{s}(\ell)$ where the entry is one, and the remaining entries are equal to zero. We treat these two cases separately. Along these lines, we introduce the convenient notation $\mathcal{S}_\ell^1 = \{k : \mathbf{s}(\ell, k) = 1\}$ and $\mathcal{S}_\ell^0 = \{k : \mathbf{s}(\ell, k) = 0\}$. This partition delineates how we approach the expectation and, ultimately, what quantities need to be evaluated through numerical methods. Recall that the state estimate for each PME element has the form

$$\hat{s}_\ell\left(q, r, \tau\right) = \frac{q e^{-\frac{(r-d_\ell)^2}{2\tau^2}}}{q e^{-\frac{(r-d_\ell)^2}{2\tau^2}} + (1-q)e^{-\frac{r^2}{2\tau^2}}}, \tag{51}$$

where $q = 1-(1-1/m_\ell)^{K_{\rm a}} \approx K_{\rm a}/m_\ell$. For a specific section, we can then write

$$\begin{aligned}&\mathbb{E}\left[\left\|\hat{\mathbf{s}}_\ell\left(d_\ell \mathbf{s}(\ell) + \tau_t\boldsymbol{\zeta}_t(\ell), \tau_t\right) - \mathbf{s}(\ell)\right\|^2\right] \\ &= \sum_{k\in\mathcal{S}_\ell^1} \mathbb{E}\left[\left(\hat{s}_\ell\left(q, d_\ell + \tau_t\boldsymbol{\zeta}_t(\ell,k), \tau_t\right) - 1\right)^2\right] \\ &\quad + \sum_{k\in\mathcal{S}_\ell^0} \mathbb{E}\left[\left(\hat{s}_\ell\left(q, \tau_t\boldsymbol{\zeta}_t(\ell,k), \tau_t\right) - 0\right)^2\right] \\ &= K_{\rm a}\mathbb{E}\left[\left(\frac{q e^{-\frac{(\tau_t\zeta)^2}{2\tau_t^2}}}{q e^{-\frac{(\tau_t\zeta)^2}{2\tau_t^2}} + (1-q)e^{-\frac{(d_\ell+\tau_t\zeta)^2}{2\tau_t^2}}} - 1\right)^2\right] \\ &\quad + (m_\ell - K_{\rm a})\mathbb{E}\left[\left(\frac{q e^{-\frac{(\tau_t\zeta-d_\ell)^2}{2\tau_t^2}}}{q e^{-\frac{(\tau_t\zeta-d_\ell)^2}{2\tau_t^2}} + (1-q)e^{-\frac{(\tau_t\zeta)^2}{2\tau_t^2}}}\right)^2\right] \\ &= K_{\rm a}\mathbb{E}\left[\left(\frac{(1-q)e^{\frac{d_\ell\zeta}{\tau_t}}}{q e^{\frac{d_\ell^2}{2\tau_t^2}} + (1-q)e^{\frac{d_\ell\zeta}{\tau_t}}}\right)^2\right] \\ &\quad + (m_\ell - K_{\rm a})\mathbb{E}\left[\left(\frac{q e^{\frac{d_\ell\zeta}{\tau_t}}}{q e^{\frac{d_\ell\zeta}{\tau_t}} + (1-q)e^{\frac{d_\ell^2}{2\tau_t^2}}}\right)^2\right].\end{aligned}$$

Both expectations in this sum can be evaluated via Monte Carlo simulation or through numerical integration techniques. The resulting values can subsequently be employed in conjunction with (50) to get numerical approximations for the state evolution in (48) and (49). The simplicity of this case

stems from the fact that only local information is employed in the denoiser, as depicted in Fig. 6. This notional diagram can be compared with the situation associated with the dynamic PME denoiser, which we turn to next.

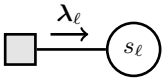

Fig. 6. This illustration shows the local neighborhood of s_ℓ under the original PME denoiser.

b) Dynamic PME Denoiser: In spirit, the state evolution equation for the dynamic PME denoiser is similar to the scenario above, except for vector $\mathbf{q}$ which is obtained from extrinsic information through belief propagation on the factor graph of the outer code. Partly motivated by the fact that the dynamic PME denoiser is Lipschitz when the number of composite iterations performed on the factor graph of the outer code is one (see Appendix C), we restrict our treatment of the state evolution to this specific case; extending this analysis to accommodate multiple iterations on the outer code is nontrivial. For the scenario of interest, the governing section equation can be written as

$$\begin{aligned}
&\mathbb{E}\left[\left\|\hat{\mathbf{s}}_\ell^{\mathrm{PME}}\left(\mathbf{Ds}+\tau_t\boldsymbol{\zeta},\tau_t\right)-\mathbf{s}(\ell)\right\|^2\right] \\
&=\sum_{k\in\mathcal{S}_\ell^1}\mathbb{E}\left[\left(\hat{s}_\ell\left(\mathbf{q}(\ell,k),d_\ell+\tau_t\boldsymbol{\zeta}_t(\ell,k),\tau_t\right)-1\right)^2\right] \\
&\quad+\sum_{k\in\mathcal{S}_\ell^0}\mathbb{E}\left[\left(\hat{s}_\ell\left(\mathbf{q}(\ell,k),\tau_t\boldsymbol{\zeta}_t(\ell,k),\tau_t\right)-0\right)^2\right] \\
&=\sum_{k\in\mathcal{S}_\ell^1}\mathbb{E}\left[\left(\frac{\left(1-\mathbf{q}(\ell,k)\right)e^{\frac{d_\ell\boldsymbol{\zeta}_t(\ell,k)}{\tau_t}}}{\mathbf{q}(\ell,k)e^{\frac{d_\ell^2}{2\tau_t^2}}+\left(1-\mathbf{q}(\ell,k)\right)e^{\frac{d_\ell\boldsymbol{\zeta}_t(\ell,k)}{\tau_t}}}\right)^2\right] \\
&\quad+\sum_{k\in\mathcal{S}_\ell^0}\mathbb{E}\left[\left(\frac{\mathbf{q}(\ell,k)e^{\frac{d_\ell\boldsymbol{\zeta}_t(\ell,k)}{\tau_t}}}{\mathbf{q}(\ell,k)e^{\frac{d_\ell\boldsymbol{\zeta}_t(\ell,k)}{\tau_t}}+\left(1-\mathbf{q}(\ell,k)\right)e^{\frac{d_\ell^2}{2\tau_t^2}}}\right)^2\right].
\end{aligned} \tag{52}$$

Recall that $\mathbf{q}(\ell,k)=1-\left(1-\frac{\boldsymbol{\mu}_{s_\ell}(k)}{\|\boldsymbol{\mu}_{s_\ell}\|_1}\right)^{K_{\mathrm{a}}}$, where $\boldsymbol{\mu}_{s_\ell}(k)=\prod_{a\in N(s_\ell)}\boldsymbol{\mu}_{a\to s_\ell}(k)$ first appears in (20). In computing $\boldsymbol{\mu}_{s_\ell}(k)$, the connectivity of the factor graph matters. For the class of triadic designs described in Section IV-C, information blocks and parity blocks must be treated separately because the structures of their local neighborhoods differ. These local neighborhoods are depicted in Fig. 7. When ℓ is a parity section, $N(s_\ell)$ is a singleton because parity sections are connected to only one check node. On the other hand, when ℓ is an information section, then $N(s_\ell)$ has a cardinality of two since every information fragment is attached to two check

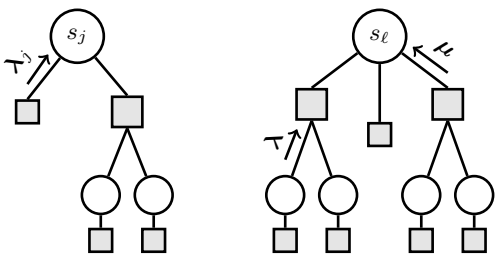

Fig. 7. For the triadic outer code design considered in this section, there are two types of local neighborhoods. The diagram on the left represent the truncated tree associated with a parity section, whereas the one of the right represents the truncated tree of an information block.

nodes. For triadic designs and in view of (20), we have

$$\begin{aligned}
\boldsymbol{\mu}_{s_\ell}(k)&=\prod_{a\in N(s_\ell)}\prod_{s_j\in N(a)\setminus s_\ell}\boldsymbol{\mu}_{s_j\to a}(k_j) \\
&=\prod_{a\in N(s_\ell)}\sum_{k_1+k_2\equiv k}\boldsymbol{\lambda}_{j_1}^{\mathrm{PME}}(k_1)\boldsymbol{\lambda}_{j_2}^{\mathrm{PME}}(k_2) \\
&=\prod_{a\in N(s_\ell)}\sum_{k_1+k_2\equiv k}\hat{s}_{j_1}\left(q,\hat{\mathbf{r}}(j_1,k_1),\tau\right)\hat{s}_{j_2}\left(q,\hat{\mathbf{r}}(j_2,k_2),\tau\right) \\
&=\prod_{a\in N(s_\ell)}\sum_{k_1+k_2\equiv k}\hat{s}_{j_1}\left(q,\mathbf{Ds}(j_1,k_1)+\tau\boldsymbol{\zeta}(j_1,k_1),\tau\right)\times \\
&\qquad\qquad\hat{s}_{j_2}\left(q,\mathbf{Ds}(j_2,k_2)+\tau\boldsymbol{\zeta}(j_2,k_2),\tau\right)
\end{aligned} \tag{53}$$

where j_1,j_2 are implicitly dependent on a. Specifically, they denote the levels that form a triad with s_ℓ through check node a within the factor graph induced by the outer code; that is, $\{j_1,j_2\}=N(a)\setminus s_\ell$. Making this relation explicit in (53) leads to an overly cumbersome notation, and context should prevent any confusion.

The components that form the left-most product in (53) are slightly involved, owing to the mixing phenomenon illustrated in Section IV-C. There are K_{a}^2 entries[4] in vector $\boldsymbol{\mu}_{s_\ell}$, including the K_{a} locations in $\mathcal{S}_\ell^1$, for which the product in (53) contains: one dominant term associated with the pair $\mathbf{s}(j_1,k_1)=\mathbf{s}(j_2,k_2)=1$; $2(K_{\mathrm{a}}-1)$ terms for which either $\mathbf{s}(j_1,k_1)=1$ or $\mathbf{s}(j_2,k_2)=1$, but not both; and the last $m_\ell-2K_{\mathrm{a}}+1$ terms for which both $\mathbf{s}(j_1,k_1)=\mathbf{s}(j_2,k_2)=0$. Furthermore, for the remaining $m_\ell-K_{\mathrm{a}}^2$ entries in $\boldsymbol{\mu}_{s_\ell}$, there are: $2K_{\mathrm{a}}$ locations where either $\mathbf{s}(j_1,k_1)=1$ or $\mathbf{s}(j_2,k_2)=1$, but not both; and $m_\ell-2K_{\mathrm{a}}$ locations where both $\mathbf{s}(j_1,k_1)=\mathbf{s}(j_2,k_2)=0$. Unfortunately, these equations do not lend themselves to compact, closed-form expressions. Moreover, the number of random variables involved in computing these quantities precludes the direct application of numerical integration. While complex and computationally demanding, it is possible to numerically evaluate the expectations in (52) through Monte-Carlo simulations using (53) and the above characterization. This requires simulating all the relevant Gaussian random variables $\boldsymbol{\zeta}(j,k)$ and computing the priors $\mathbf{q}(j,k)$ at once using the fast transform technique. Altogether, the latter approach provides a blueprint on how to iteratively compute the parameters in (49)

[4]This characterization assumes that $K_{\mathrm{a}}^2\le m_\ell$; while our framework remains valid in spirit when $K_{\mathrm{a}}^2>m_\ell$, a more careful accounting needs to be performed in this alternate regime.

and (48) with good accuracy for the dynamic PME denoiser. In Section V, we demonstrate that the performance of CCS-AMP predicted by the state evolution using the aforementioned approach is very close to the empirical performance obtained by simulating the actual system.

V. Simulation Results and Discussion

In this section, we study the empirical performance of the proposed scheme and provide comparisons with the original AMP-based algorithm introduced in [5]. For these simulations, we consider a system with $K_{\mathrm{a}} \in [10 : 300]$ active users. The size of the payload corresponding to every active user is $w = 128$ bits. The total number of channel uses is $n = 38400$. The length of each block is equal to 16 bits, i.e., $v_\ell = 16$ for all sections $\ell \in [L]$, and the number of sections $L = 16$ when $K_{\mathrm{a}} < 200$ and 18 when $K_{\mathrm{a}} \geq 200$. The target per-user probability of error is five percent, $P_{\mathrm{e}} = 0.05$. It is worth mentioning that these system parameters have become emblematic of recent contributions related to unsourced, uncoordinated random access.

Figure 5 shows the factor graph for the outer code employed in simulations when $K_{\mathrm{a}} < 200$. We add two additional parity sections to reduce the probability of the decoder producing a list of size greater than K_{a} when $K_{\mathrm{a}} \geq 200$. These architectures offer very good performance for the parameters of interest, yet we remark that our framework supports many alternate implementations worthy of investigation, possibly, in future studies. During simulations, sensing matrix $\mathbf{A} \in \mathbb{R}^{n \times m}$ is formed by picking n rows uniformly at random from a Hadamard matrix of dimension $m \times m$. This Hadamard approach reduces the memory and computational load of the AMP decoder significantly over random Gaussian constructions, owing to the fast transform implementation. Specifically, the computational complexity corresponding to one iteration of AMP can be reduced from $\mathcal{O}(nm)$ to $\mathcal{O}(m \log m)$ when a sub-sampled Hadamard matrix is utilized as the sensing matrix. Although this constitutes a small departure from our theoretical analysis, we emphasize that this approach is frequently employed in practice, and it is known to perform well in finite block length regimes [5], [15].

Another common implementation twist we incorporate into our numerical study is running the overall algorithm twice, in the spirit of successive interference cancellation. That is, we perform one extended round of AMP iterations, whereby the contributions of users decoded with high confidence are removed from the received signal, and then the residual signal is fed back into the AMP decoder for an additional round of composite iterations followed by disambiguation. To this end, we remove the contribution of the top $K_{\mathrm{a}} - \delta$ decoded messages with the largest likelihoods obtained during the first round, where δ is chosen empirically. The second round of decoding then focuses on the recovery of the remaining δ messages from the residual signal. Finally, if the decoder produces a list of size greater than K_{a}, we retain the K_{a} messages with the largest likelihoods.

A prime goal of the numerical section is to demonstrate that the enhanced CCS-AMP algorithm described in this article

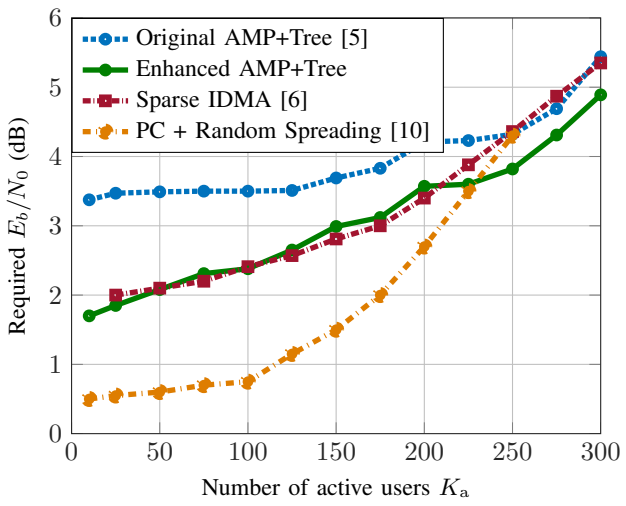

Fig. 8. The graph compares the performance of the proposed scheme with that of other existing schemes.

delivers significant improvements over the original algorithm introduced by Fengler et al. [5] without excessively increasing the computational load. Figure 8 shows the minimum energy-per-bit necessary to achieve the per user probability of error as a function of the number of active devices. Formally, the *energy-per-bit* of the system is defined as $\frac{E_b}{N_0} = \frac{nP}{2w}$, where P denotes the transmit symbol power. The top curve therein demonstrates the performance of the original AMP-based scheme introduced in [5], which essentially employs uninformative prior probabilities within every AMP composite iteration. The performance of the CCS-AMP algorithm that dynamically leverages belief propagation on the factor graph of the outer code is captured by the solid curve. It can be seen that the enhanced decoder outperforms the original decoder for all values of K_{a}, and the gain is more pronounced for small values of K_{a}. Furthermore, the numerical simulations feature the same parameters, the same underlying factor graphs, sensing matrices, and two-pass decoding. In this sense, the comparison is straightforward and fair. The third curve, which is based on sparse-IDMA [6] represents the state-of-the-art for $K_{\mathrm{a}} \geq 250$. Our proposed scheme outperforms this benchmark in this same region and exhibits comparable performance in other regions. The bottom most curve corresponds to a polar coding and random spreading based scheme [10] which is the state-of-the-art for $K_{\mathrm{a}} \leq 250$. This latter scheme outperforms our proposed scheme for small values of K_{a}, but falls behind when $K_{\mathrm{a}} > 225$. It is also worth mentioning that the polar coding scheme is computationally much more demanding. This may prevent its applications in certain scenarios, especially in real-time settings.

Another important approach to highlight the benefits of enhanced CCS-AMP over the original AMP-based scheme is through the state evolution discussed in Section IV-D. Recall that parameter τ_t^2 can be interpreted as the variance of the noise in the effective observation at AMP iteration t. A system is performing better when this quantity decreases rapidly as a function of the iteration count, and when its

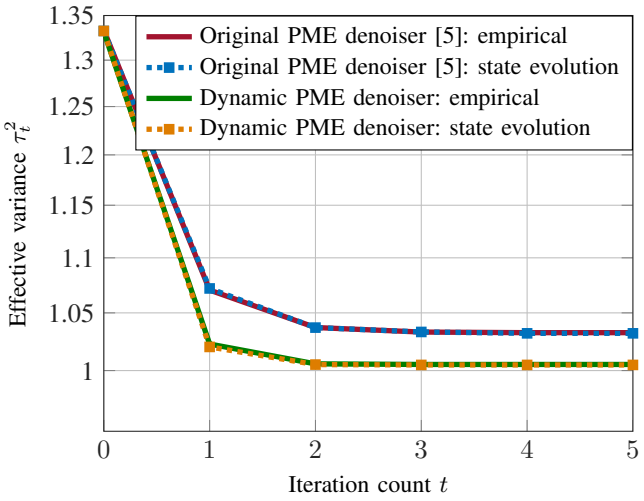

Fig. 9. The figure compares the variance parameter τ_t^2 obtained empirically with that predicted by the state evolution for the original PME denoiser in [5] and the proposed dynamic PME denoiser. The parameters used to generate these plots are: $K_{\mathrm{a}} = 25, E_b/N_0 = 3\ dB$.

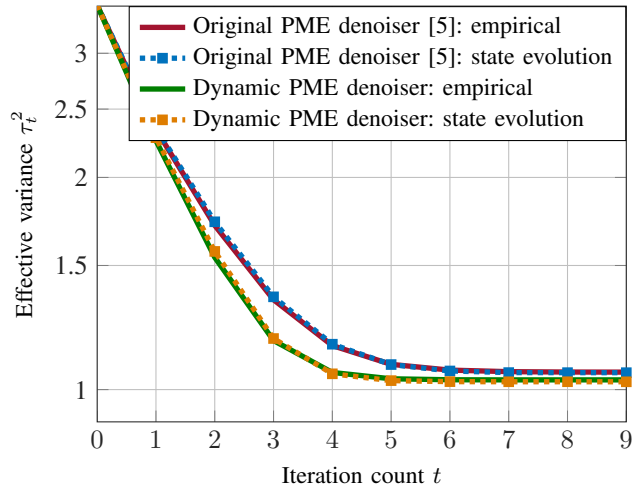

Fig. 10. This plot compares the variance parameter τ_t^2 obtained empirically with that predicted by state evolution for the original PME denoiser in [5] and the proposed dynamic PME denoiser. The parameters used to generate these plots are: $K_{\mathrm{a}} = 150, E_b/N_0 = 4\ dB$.

asymptotic value is low. In Fig. 9 and Fig. 10, we provide a comparison of the two schemes using this viewpoint. This is accomplished by plotting the variance parameters for the two algorithms obtained empirically through numerical simulations and via the state evolution framework described in Section IV-D for $K_{\mathrm{a}} = 25$ and $K_{\mathrm{a}} = 150$, respectively. In both cases, the enhanced CCS-AMP algorithm offers noticeable improvements in terms of decay rate and asymptotic value. We also note that the empirical simulations are very close to the curves afforded by our asymptotic analysis. This suggests that the state evolution accurately predicts system performance for both the original PME denoiser in [5] and the dynamic PME denoiser put forth in this article, thereby validating the framework developed in Section IV-D. In other words, the system parameters we are interested in are well within the regime where the performance is predicted accurately by the state evolution.

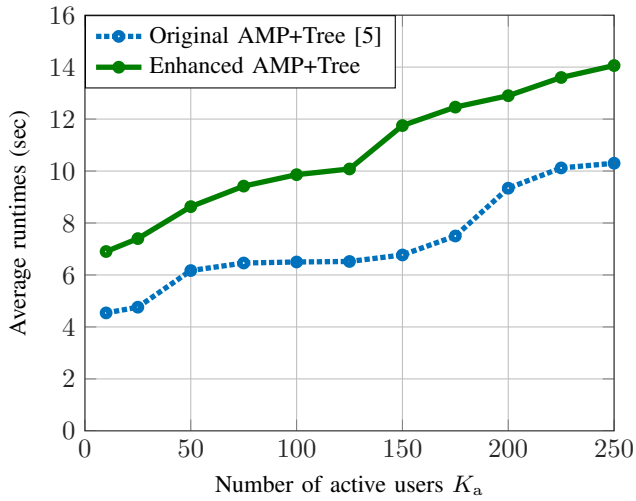

Fig. 11. This plot compares the average run-time of the proposed scheme with the scheme in [5].

Figure 11 compares the runtimes of the proposed decoder and the original decoder in [5]. A recurring theme that underlies much of the research in unsourced, uncoordinated random access is that performance gain often comes at the cost of computational complexity. It is therefore pertinent to discuss the computational burden imposed by the dynamic PME denoiser. We can see from this graph that runtimes are comparable, although the enhanced version does require more resources. Overall, it seems that for most applications, the performance benefits afforded by the enhanced version would outweigh the computational overhead created by the message passing over the outer factor graph.

VI. Conclusion

This article presents a novel algorithm for unsourced, uncoordinated random access. The proposed scheme builds on the connection between coded compressed sensing (CCS) and approximate message passing (AMP), which was first identified by Fengler et al. in [5]. A main contribution of our work is the realization that the inner and outer codes embedded in this connection can be made to interact dynamically within a unified iterative framework. Yet, making this interactive framework possible demands several key innovations. The design of the outer tree code has to be modified in a way that enables belief propagation on a factor graph, while also providing meaningful information to the AMP inner code. To this end, the class of codes considered herein are based on triadic designs. Second, parity blocks have to be created in a manner conducive to the application of fast Fourier techniques or equivalent algorithms. This is achieved by making sure that the parity precursors have a circular convolution structure. The dynamic approach yields a non-separable denoiser that is amenable to analysis through the state evolution, and the overall performance is a significant improvement over the original AMP-based scheme.

The framework introduced in the article also points to several possible avenues moving forward. The dynamic interaction between the AMP inner code and the BP-based outer code seems natural. Although the architecture presented in the article is rooted in the lessons learned from previous contributions, it seems that the framework can be extended to a rich class of factor graphs for the outer code. While a circular structure conducive to FFT techniques is used in our treatment of the algorithm, the proposed paradigm also extend to other groups or fields. For instance, the Walsh–Hadamard transform can also be utilized as a means to efficiently compute beliefs on a different factor graph. This discussion points to possible improvements in performance based on alternate designs for the outer code, possibly exploring hierarchical or cascading structures. Likewise, there are options to consider with respect to the design of the sensing matrix. In particular, the design philosophy put forth in the article could be retrofitted to coded compressed sensing as a means to either devise a system with good performance and lower complexity, or as a strategy to handle observation vector that are much larger.

Though not immediately obvious, it may be possible to extend the CCS-AMP scheme proposed in this article to realistic scenarios involving quasi-static block fading channels. In such scenarios, the decoding algorithm should take into account the prior distribution of the fade levels in designing a denoiser that is amenable to belief propagation. We envision that such an implementation could lead to a joint data-channel estimation scheme. These and other research avenues are beyond the scope of this article, and they are left as possible future endeavors.

Appendix A
Performance of Tree Code Revisited

The performance of the original tree code, as it pertains to coded compressed sensing, is studied at length in [4]. Herein, we show that, although the parity encoding is fundamentally different for the alternate outer code, the structure of the code and the ensuing statistical properties that underlie overall performance are preserved. Recall that the difference between the original tree code and the revised outer code lies in the generation of parity bits. In the original scheme, parities are created using random linear combinations of information bits

$$\mathbf{p}'(\ell) = \sum_{j=1}^{\ell-1} \mathbf{w}(j)\mathbf{G}_{j,\ell} \tag{54}$$

whereas, in the current scheme, we have

$$\mathbf{p}(\ell) = f_{\mathbb{Z}/2^{p\ell}\mathbb{Z}\to\mathbb{F}_2^{p\ell}}\left(\sum_{j=1}^{\ell-1} f_{\mathbb{F}_2^{p\ell}\to\mathbb{Z}/2^{p\ell}\mathbb{Z}}\big(\mathbf{w}(j)\mathbf{G}_{j,\ell}\big) \mod 2^{p\ell}\right). \tag{55}$$

Beyond this distinction, the two outer codes are structurally identical whenever acting on a same factor graph. Below, we show that the statistical properties that serve as a foundation for the analysis of tree decoding in [4] are preserved under the alternate parity generation of (55).

A. Single-Fragment Message

Paralleling the development in [4], we first examine the case where $\mathbf{v}$ features a single fragment, as depicted in Fig. 12. Specifically, consider a binary message $\mathbf{w}$ of length w. Parity bits are generated for this message using (55) with generator matrix $\mathbf{G}$. The ensuing codeword, $\mathbf{v} = \mathbf{wp}$, is then created by taking message $\mathbf{w}$ and appending parity vector $\mathbf{p}$ to it, with

$$\begin{aligned}\mathbf{p} &= f_{\mathbb{Z}/2^p\mathbb{Z}\to\mathbb{F}_2^p}\left(f_{\mathbb{F}_2^p\to\mathbb{Z}/2^p\mathbb{Z}}\big(\mathbf{wG}\big) \mod 2^p\right)\\ &= f_{\mathbb{Z}/2^p\mathbb{Z}\to\mathbb{F}_2^p}\left(f_{\mathbb{F}_2^p\to\mathbb{Z}/2^p\mathbb{Z}}\big(\mathbf{wG}\big)\right) = \mathbf{wG}.\end{aligned}$$

Since the alternate encoding is equivalent to the original encoding for the one-fragment case, the revised outer encoding trivially inherits the following result.

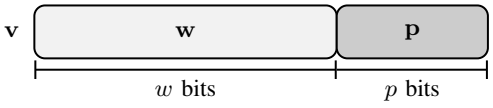

Fig. 12. Message $\mathbf{v}$ is obtained by starting with message fragment $\mathbf{w}$ and appending p parity bits to it.

Lemma 11. *Fix information vector* $\mathbf{w}$ *and parity generating matrix* $\mathbf{G}$. *The probability that a randomly selected information vector* $\mathbf{w}_\mathrm{r} \in \{0,1\}^w$ *produces the same parity subcomponent as* $\mathbf{w}$ *under* $\mathbf{G}$ *is given by*

$$\Pr(\mathbf{p} = \mathbf{p}_\mathrm{r}) = 2^{-\operatorname{rank}(\mathbf{G})}.$$

A quantity of interest to our upcoming discussion is the probability distributions of parity patterns for distinct information messages. Because of the non-linear operations involved in creating parity bits, through multiple representations, the derivation of these distributions is admittedly cumbersome, yet necessary.

Lemma 12. *Fix erroneous vector* $\mathbf{w}_\mathrm{e} \neq \mathbf{w}$. *Let parity generator matrix* $\mathbf{G}$ *be a Rademacher matrix of size* $w \times p$. *That is, the entries in* $\mathbf{G}$ *are drawn at random from a uniform Bernoulli distribution, independently of one another. Under such circumstances,*

$$f_{\mathbb{F}_2^p\to\mathbb{Z}/2^p\mathbb{Z}}\big(\mathbf{wG}\big) - f_{\mathbb{F}_2^p\to\mathbb{Z}/2^p\mathbb{Z}}\big(\mathbf{w}_\mathrm{e}\mathbf{G}\big) \mod 2^p \tag{56}$$

is uniformly distributed over $\mathbb{Z}/2^p\mathbb{Z}$ *and the probability of event* $\{\mathbf{p} = \mathbf{p}_\mathrm{e}\}$ *is equal to*

$$\Pr(\mathbf{p} = \mathbf{p}_\mathrm{e}) = 2^{-p}.$$

Proof: The event $\{\mathbf{p} = \mathbf{p}_\mathrm{e}\}$ is equivalent to $\{\mathbf{wG} = \mathbf{w}_\mathrm{e}\mathbf{G}\}$. Since $\mathbf{w} \neq \mathbf{w}_\mathrm{e}$, there exists at least one pair of vector entries, say at location i, such that $(\mathbf{w})_i \neq (\mathbf{w}_\mathrm{e})_i$. In view of the symmetry in the problem, we can assume without loss of generality that $(\mathbf{w})_i = 1$ and $(\mathbf{w}_\mathrm{e})_i = 0$. Then, $\mathbf{w}_\mathrm{e}\mathbf{G} = \sum_{r\neq i}(\mathbf{w}_\mathrm{e})_r\mathbf{G}[r,:]$; we emphasize that the ith row of $\mathbf{G}$ does not enter this summation. For any $k \in \mathbb{Z}/2^p\mathbb{Z}$, we can express the probability of the event of interest as in (57). By constructions, $\mathbf{G}[i,:]$ is a sequence of uniform Bernoulli trials, independent of the term on the right-hand-side. Therefore, the probability that this equality is met is 2^{-p}, irrespective of k.

$$\begin{aligned}
&\Pr\left(f_{\mathbb{F}_2^p\to\mathbb{Z}/2^p\mathbb{Z}}\big(\mathbf{w}\mathbf{G}\big)-f_{\mathbb{F}_2^p\to\mathbb{Z}/2^p\mathbb{Z}}\big(\mathbf{w}_{\rm e}\mathbf{G}\big)\equiv k \mod 2^p\right)\\
&=\Pr\left(f_{\mathbb{F}_2^p\to\mathbb{Z}/2^p\mathbb{Z}}\big(\mathbf{w}\mathbf{G}\big)\equiv k+f_{\mathbb{F}_2^p\to\mathbb{Z}/2^p\mathbb{Z}}\left(\sum_{r\neq i}(\mathbf{w}_{\rm e})_r\mathbf{G}[r,:]\right) \mod 2^p\right)\\
&=\Pr\left(\mathbf{w}\mathbf{G}=f_{\mathbb{Z}/2^p\mathbb{Z}\to\mathbb{F}_2^p}\left(k+f_{\mathbb{F}_2^p\to\mathbb{Z}/2^p\mathbb{Z}}\left(\sum_{r\neq i}(\mathbf{w}_{\rm e})_r\mathbf{G}[r,:]\right) \mod 2^p\right)\right)\\
&=\Pr\left(\mathbf{G}[i,:]=f_{\mathbb{Z}/2^p\mathbb{Z}\to\mathbb{F}_2^p}\left(k+f_{\mathbb{F}_2^p\to\mathbb{Z}/2^p\mathbb{Z}}\left(\sum_{r\neq i}(\mathbf{w}_{\rm e})_r\mathbf{G}[r,:]\right) \mod 2^p\right)-\sum_{r\neq i}(\mathbf{w})_r\mathbf{G}[r,:]\right)
\end{aligned} \tag{57}$$

That is, (56) is uniformly distributed over $\mathbb{Z}/2^p\mathbb{Z}$, as claimed. The second part of the lemma follows immediately from above and the observation that the event $\{\mathbf{p}=\mathbf{p}_{\rm e}\}$ is true if and only if (56) is congruent to zero in $\mathbb{Z}/2^p\mathbb{Z}$. ∎

B. Multi-Fragment Message

The situation becomes slightly more complicated when parity bits are generated over multiple fragments. This produces a structure akin to the one portrayed in Fig. 13. Every parity pattern $\mathbf{p}(\ell)$ therein is produced using (55). We wish to understand how this new way of adding redundancy impacts the probability of erroneous paths staying alive as the outer decoder proceeds forward through the various levels during message disambiguation. To begin, we look at the probability that an erroneous (partial) path produces the same parity pattern as a valid codeword for one block.

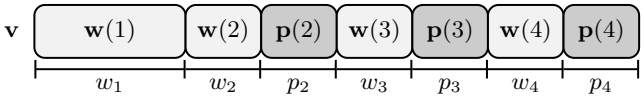

Fig. 13. The fragmented nature of outer coding leads to a certain structure in $\mathbf{v}$, as depicted above. This renders the performance analysis of this scheme more challenging due, partly, to the fact that two different messages can share identical information fragments.

Lemma 13. *Suppose information message* $\mathbf{w}$ *is fixed. Consider the truncated erroneous vector*

$$\mathbf{w}_{\rm e}(1)\cdots\mathbf{w}_{\rm e}(\ell-1)\neq\mathbf{w}(0)\cdots\mathbf{w}(\ell-1). \tag{58}$$

Let $\{\mathbf{G}_{j,\ell}\}$ *be a collection of independent Rademacher matrices, each of size* $w_j\times p_\ell$*. In other words, the entries in* $\mathbf{G}_{j,\ell}$ *are drawn at random from a uniform Bernoulli distribution, independently of one another and of other matrices. Under such circumstances, the probability of event* $\{\mathbf{p}=\mathbf{p}_{\rm e}\}$ *is given by*

$$\Pr(\mathbf{p}(\ell)=\mathbf{p}_{\rm e}(\ell))=2^{-p_\ell}.$$

Proof: Within the revised outer encoding, parity bits at level ℓ are generated using (55). Accordingly, $\mathbf{p}(\ell)=\mathbf{p}_{\rm e}(\ell)$ if and only if

$$\begin{aligned}
&\sum_{j=1}^{\ell-1}f_{\mathbb{F}_2^{p_\ell}\to\mathbb{Z}/2^{p_\ell}\mathbb{Z}}\big(\mathbf{w}(j)\mathbf{G}_{j,\ell}\big)\\
&\equiv\sum_{j=1}^{\ell-1}f_{\mathbb{F}_2^{p_\ell}\to\mathbb{Z}/2^{p_\ell}\mathbb{Z}}\big(\mathbf{w}_{\rm e}(j)\mathbf{G}_{j,\ell}\big) \mod 2^{p_\ell}.
\end{aligned} \tag{59}$$

This follows because $f_{\mathbb{Z}/2^{p_\ell}\mathbb{Z}\to\mathbb{F}_2^{p_\ell}}(\cdot)$ is a bijection. Since truncated vectors $\mathbf{w}_{\rm e}(1)\cdots\mathbf{w}_{\rm e}(\ell-1)\neq\mathbf{w}(1)\cdots\mathbf{w}(\ell-1)$, there exists a k such that $\mathbf{w}_{\rm e}(k)\neq\mathbf{w}(k)$. With this k, we can rewrite (59) as

$$\begin{aligned}
&\Big(f_{\mathbb{F}_2^{p_\ell}\to\mathbb{Z}/2^{p_\ell}\mathbb{Z}}\big(\mathbf{w}(k)\mathbf{G}_{k,\ell}\big)-f_{\mathbb{F}_2^{p_\ell}\to\mathbb{Z}/2^{p_\ell}\mathbb{Z}}\big(\mathbf{w}_{\rm e}(k)\mathbf{G}_{k,\ell}\big)\Big)\\
&+\sum_{\substack{j=1\\ j\neq k}}^{\ell-1}\Big(f_{\mathbb{F}_2^{p_\ell}\to\mathbb{Z}/2^{p_\ell}\mathbb{Z}}\big(\mathbf{w}(j)\mathbf{G}_{j,\ell}\big)-f_{\mathbb{F}_2^{p_\ell}\to\mathbb{Z}/2^{p_\ell}\mathbb{Z}}\big(\mathbf{w}_{\rm e}(j)\mathbf{G}_{j,\ell}\big)\Big)\\
&\equiv 0 \mod 2^{p_\ell}.
\end{aligned} \tag{60}$$

By construction, the isolated difference and random matrix $\mathbf{G}_{k,\ell}$ are independent of the succeeding summation. Furthermore, Lemma 12 asserts that this first term is uniformly distributed over $\mathbb{Z}/2^{p_\ell}\mathbb{Z}$. We then deduce that, under condition (58), $\Pr\left(\mathbf{p}(\ell)=\mathbf{p}_{\rm e}(\ell)\right)=2^{-p_\ell}$, as desired. ∎

At this point, we draw a parallel between the properties of the revised outer code and those of the original tree code. There remains a dichotomy in the impact of parity patterns: the parity section $\mathbf{p}(\ell)$ associated with stage ℓ either acts as a statistically discriminating sequence of independent Bernoulli samples, each with probability half, or the parity conditions are fulfilled trivially when the truncated vectors match. Beyond this observation, there remain three confounding factors in the analysis of multi-fragment codewords. First, several fragments within an erroneous candidate codeword may come from a same message; overlapping fragments reduce the propensity for parity bits to be statistically discriminating. Second, two different messages may have identical information fragments, as these messages only need to differ in one location overall. Mathematically, when comparing a valid codeword to an erroneous candidate, the two messages are necessarily distinct with $\mathbf{w}_{\rm e}\neq\mathbf{w}$; however, it is possible to have $\mathbf{w}_{\rm e}(\ell)=\mathbf{w}(\ell)$ for a (strict) subset of $\{1,\ldots,L\}$. Finally, the loss of discriminating power from parity constraints may be correlated

across fragments in certain cases, which exacerbates the error probability. Nevertheless, these confounding factors seem intrinsic to the outer code structure. They remain unchanged irrespective of whether parity bits are constructed using the original scheme or the revised method. Consequently, these two distinct approaches yield identical expected performance, when applied to a same factor graph.

Proposition 14. *The average performance of the revised outer code with the parity structure of* (55) *and that of the original tree code introduced in [4], [29] with random linear combinations as in* (54) *are identical, when acting on a same factor graph.*

Proof: Consider a situation where the outer decoder seeks to validate codewords that start with root fragment $\mathbf{w}_{i_1}(1)$. For a given collection of K_{a} transmitted codewords, the list of candidate codewords visited during this phase of the disambiguation process is composed of elements of the form

$$\begin{aligned}\mathbf{v}_{\mathrm{c}} &= \mathbf{v}_{i_1}(1)\mathbf{v}_{i_2}(2)\cdots\mathbf{v}_{i_L}(L)\\ &= \mathbf{w}_{i_1}(1)\mathbf{w}_{i_2}(2)\mathbf{p}_{i_2}(2)\cdots\mathbf{w}_{i_L}(L)\mathbf{p}_{i_L}(L),\end{aligned} \tag{61}$$

where $i_\ell \in [1:K_{\mathrm{a}}]$ for all slots $\ell \in [2:L]$. The ability of the outer decoder to identify an erroneous sequence hinges on the structure of the candidate vector. In particular, this erroneous codeword will be inconspicuous if the parity patterns are consistent, i.e.,

$$\begin{aligned}&\mathbf{p}_{i_\ell}(\ell)\\ &= f_{\mathbb{Z}/2^{p_\ell}\mathbb{Z}\to\mathbb{F}_2^{p_\ell}}\left(\sum_{j=1}^{\ell-1} f_{\mathbb{F}_2^{p_\ell}\to\mathbb{Z}/2^{p_\ell}\mathbb{Z}}\big(\mathbf{w}_{i_j}(j)\mathbf{G}_{j,\ell}\big) \mod 2^{p_\ell}\right)\\ &\quad \ell = 2,\ldots,L.\end{aligned} \tag{62}$$

In view of Lemma 13 and for a fixed index sequence $i_1, i_2, \ldots, i_L$, the probability of the event above is equal to the probability of the erroneous sequence being undetected under the original tree code, i.e.,

$$\mathbf{p}'_{i_\ell}(\ell) = \sum_{j=1}^{\ell-1}\mathbf{w}_{i_\ell}(j)\mathbf{G}_{j,\ell} = \sum_{j=1}^{\ell-1}\mathbf{w}_{i_j}(j)\mathbf{G}_{j,\ell} \quad \ell = 2,\ldots,L. \tag{63}$$

That is, the probability that an erroneous sequence survives, conditioned on a given index sequence, is the same in both systems. Because this statement holds for every possible index sequence, the analysis of performance in [4], which relies on the groupings of such conditional probabilities, too applies for the revised outer code. That is, performance in terms of the expected number of surviving paths and overall probability of decoding failure is identical for the two schemes. ∎

We conclude this section with a brief remark. We demonstrate above that, for a given factor graph, the revised outer code offers the same performance as the original tree code. Yet, the revised version enables the possibility of using fast Fourier techniques. This fact is significant in that it ensures that the proposed design is sound. For instance, in theory, the revised outer code could be employed to handle soft estimates (rather than hard decisions) within the original CCS scheme as well. Still, although the performance of the two approaches is identical when applied to a same factor graph, CCS-AMP ultimately adopts a different type of graph for its outer code, with a parity allocation attuned to the AMP inner code.

APPENDIX B
MESSAGE PASSING RULES ON THE GRAPH

In this section, we derive the form of the messages passed from check nodes to variable nodes on the factor graph associated with the outer code. These messages are employed in the context of belief propagation, as discussed in Section III. Paralleling our earlier treatment, we assume that the effective observation available to the outer decoder is of the form $\mathbf{r} = \mathbf{D}\mathbf{s} + \tau\boldsymbol{\zeta}$, where $\boldsymbol{\zeta}$ is an i.i.d. $\mathcal{N}(0,1)$ random vector and τ is the standard deviation of the individual noise components.

A. Local Estimates

A quantity needed for belief propagation is a local estimate for the probability that a particular device i is sending a message $\mathbf{m}_i$ whose kth component in section ℓ is a one, conditioned on effective observation $\mathbf{r}(\ell)$. As an initial step, we examine a sequence of one-sparse (labeled) candidate blocks $\hat{\mathbf{m}}_1(\ell),\ldots,\hat{\mathbf{m}}_{K_{\mathrm{a}}}(\ell)$ with $\ell \in [L]$. Under the Gaussian model, the conditional probability of this candidate sequence, given $\mathbf{r}(\ell)$, is governed by

$$\begin{aligned}&\Pr\left(\hat{\mathbf{m}}_1(\ell),\ldots,\hat{\mathbf{m}}_{K_{\mathrm{a}}}(\ell)|\mathbf{r}(\ell)\right)\\ &= \frac{\exp\left(-\frac{\|\mathbf{r}(\ell)-d_\ell\hat{\mathbf{s}}(\ell)\|^2}{2\tau^2}\right)}{\sum_{\hat{\mathbf{m}}_1(\ell),\ldots,\hat{\mathbf{m}}_{K_{\mathrm{a}}}(\ell)}\exp\left(-\frac{\|\mathbf{r}(\ell)-d_\ell\hat{\mathbf{s}}(\ell)\|^2}{2\tau^2}\right)}\end{aligned} \tag{64}$$

where $\hat{\mathbf{s}}(\ell) = \sum_{i\in[K_{\mathrm{a}}]}\hat{\mathbf{m}}_i(\ell)$ and the sum in the denominator ranges over all possible assignments. We stress that (64) relies on the assumption that message block $\mathbf{m}_i(\ell)$ is selected uniformly at random from the m_ℓ one-sparse candidates, independently from other messages. We further examine the exponent in (64) by writing

$$\begin{aligned}&\|\mathbf{r}(\ell) - d_\ell\hat{\mathbf{s}}(\ell)\|^2 = \|\mathbf{r}(\ell)\|^2 - 2d_\ell\langle\mathbf{r}(\ell),\hat{\mathbf{s}}(\ell)\rangle + d_\ell^2\|\hat{\mathbf{s}}(\ell)\|^2\\ &= \|\mathbf{r}(\ell)\|^2 - 2d_\ell\sum_{i\in[K_{\mathrm{a}}]}\langle\mathbf{r}(\ell),\hat{\mathbf{m}}_i(\ell)\rangle + d_\ell^2 K_{\mathrm{a}}\\ &\quad + d_\ell^2\left(\|\hat{\mathbf{s}}(\ell)\|^2 - K_{\mathrm{a}}\right)\\ &= \|\mathbf{r}(\ell)\|^2 + \sum_{i\in[K_{\mathrm{a}}]}\left(\left(\mathbf{r}\left(\ell,k^{(i)}\right) - d_\ell\right)^2 - \mathbf{r}\left(\ell,k^{(i)}\right)^2\right)\\ &\quad + d_\ell^2\left(\|\hat{\mathbf{s}}(\ell)\|^2 - K_{\mathrm{a}}\right)\end{aligned} \tag{65}$$

where $k^{(i)}$ is the index of the unique non-zero entry in $\hat{\mathbf{m}}_i(\ell)$. We can express (64) using this characterization,

$$\begin{aligned}&\Pr\left(\hat{\mathbf{m}}_1(\ell)\ldots,\hat{\mathbf{m}}_{K_{\mathrm{a}}}(\ell)|\mathbf{r}(\ell)\right)\\ &\propto e^{-\frac{d_\ell^2\left(\|\hat{\mathbf{s}}(\ell)\|^2-K_{\mathrm{a}}\right)}{2\tau^2}}\prod_{i\in[K_{\mathrm{a}}]}e^{-\frac{\left(\mathbf{r}\left(\ell,k^{(i)}\right)-d_\ell\right)^2-\mathbf{r}\left(\ell,k^{(i)}\right)^2}{2\tau^2}}\\ &\propto e^{-\frac{d_\ell^2\left(\|\hat{\mathbf{s}}(\ell)\|^2-K_{\mathrm{a}}\right)}{2\tau^2}}\prod_{i\in[K_{\mathrm{a}}]}\mathcal{L}_\ell\left(k^{(i)}\right)\end{aligned} \tag{66}$$

where the last line makes use of the likelihood notation $\mathcal{L}_\ell(k) = \exp\left(-\frac{(\mathbf{r}(\ell,k)-d_\ell)^2-\mathbf{r}(\ell,k)^2}{2\tau^2}\right)$. Turning to the marginal distribution of a specific device, say device one, we can calculate the probability that its message block ℓ contains a one at location $k^{(1)}$ based on (66). Specifically, we get the intricate normalized expression

$$\begin{aligned}&\Pr\left(\hat{\mathbf{m}}_1(\ell)|\mathbf{r}(\ell)\right)\\ &= \frac{\sum_{(\hat{\mathbf{m}}_2(\ell)\ldots,\hat{\mathbf{m}}_{K_\mathrm{a}}(\ell))} e^{-\frac{d_\ell^2\left(\|\hat{\mathbf{s}}(\ell)\|^2-K_\mathrm{a}\right)}{2\tau^2}} \prod_{i\in[K_\mathrm{a}]} \mathcal{L}_\ell\left(k^{(i)}\right)}{\sum_{(\hat{\mathbf{m}}_1(\ell)\ldots,\hat{\mathbf{m}}_{K_\mathrm{a}}(\ell))} e^{-\frac{d_\ell^2\left(\|\hat{\mathbf{s}}(\ell)\|^2-K_\mathrm{a}\right)}{2\tau^2}} \prod_{i\in[K_\mathrm{a}]} \mathcal{L}_\ell\left(k^{(i)}\right)}.\end{aligned} \tag{67}$$

While this characterization is exact, computing local marginal probabilities based on (67) is a formidable task for the problem dimensions we are interested in. This is impractical for the iterative algorithm we wish to develop and, consequently, we seek a suitable low-complexity approximation.

As a first step, we neglect assignments with multiplicities, i.e., $\|\hat{\mathbf{s}}(\ell)\|_0 < K_\mathrm{a}$; this seems reasonable because these cases are heavily discounted by the multiplicative factor $\exp\left(-\frac{d_\ell^2\left(\|\hat{\mathbf{s}}(\ell)\|^2-K_\mathrm{a}\right)}{2\tau^2}\right)$ and they are also unlikely when sections are large. It may be helpful to point out that, in comparison, this factor is equal to one when there are no collisions, i.e., $\|\hat{\mathbf{s}}(\ell)\|_0 = K_\mathrm{a}$. Disregarding collisions, the ratio in (67) becomes simpler. Under index notation, marginal estimates can be written as

$$\begin{aligned}\Pr\left(\mathbf{m}_1(\ell,k)=1|\mathbf{r}(\ell)\right) &\approx \frac{\mathcal{L}_\ell(k)\sum_{\boldsymbol{\kappa}:k\in\boldsymbol{\kappa}}\prod_{\kappa\in\boldsymbol{\kappa}\setminus k}\mathcal{L}_\ell(\kappa)}{K_\mathrm{a}\sum_{\boldsymbol{\kappa}}\prod_{\kappa\in\boldsymbol{\kappa}}\mathcal{L}_\ell(\kappa)}\\ &= \frac{\mathcal{L}_\ell(k)\sum_{\boldsymbol{\kappa}:k\in\boldsymbol{\kappa}}\prod_{\kappa\in\boldsymbol{\kappa}\setminus k}\mathcal{L}_\ell(\kappa)}{K_\mathrm{a}\mathcal{L}_\ell(k)\sum_{\boldsymbol{\kappa}:k\in\boldsymbol{\kappa}}\prod_{\kappa\in\boldsymbol{\kappa}\setminus k}\mathcal{L}_\ell(\kappa)+\sum_{\boldsymbol{\kappa}:k\notin\boldsymbol{\kappa}}\prod_{\kappa\in\boldsymbol{\kappa}}\mathcal{L}_\ell(\kappa)}\end{aligned} \tag{68}$$

The set $\boldsymbol{\kappa}$ represents a subset of K_a distinct indices in $\{0,\ldots,m_\ell-1\}$. Every summand accounts for collections of non-colliding assignments $\hat{\mathbf{m}}_1(\ell),\ldots,\hat{\mathbf{m}}_{K_\mathrm{a}}(\ell)$ whose equivalent index representations are jointly equal to $\boldsymbol{\kappa}$. We stress that there are $K_\mathrm{a}!$ such assignments for a given $\boldsymbol{\kappa}$, one for each index permutation. The corresponding entries in (67) all share the same value because the product of likelihoods is permutation invariant.

Computing (68) remains a sizeable task for parameters of interest, and it may be too costly to embed this step in an AMP iteration. Accordingly, we simplify our marginal estimator by replacing the likelihoods by a scaled version of the marginal posterior mean estimate (PME) introduced by Fengler et al. [5]. This estimate is discussed at length in Section IV-A. For the time being, it suffices to state that

$$\begin{aligned}&\Pr\left(\mathbf{m}_1(\ell,k)=1|\mathbf{r}(\ell)\right)\\ &\approx \frac{1}{K_\mathrm{a}}\frac{qe^{-\frac{\left(\mathbf{r}(\ell,k)-d_\ell\right)^2}{2\tau^2}}}{(1-q)e^{-\frac{\mathbf{r}(\ell,k)^2}{2\tau^2}}+qe^{-\frac{\left(\mathbf{r}(\ell,k)-d_\ell\right)^2}{2\tau^2}}}\end{aligned} \tag{69}$$

where q and τ are a judiciously selected constants. While (69) is guaranteed to produce bounded, non-negative elements; the ensuing vector may not be normalized. We can rectify this situation by adding a normalizing factor to the estimates. Altogether, after accounting for the reshuffling, we propose to use block estimate $\boldsymbol{\lambda}_\ell(k)$ as a proxy for the marginal probability $\Pr\left(\mathbf{m}_1(\ell,k)=1|\mathbf{r}(\ell)\right)$, where we define

$$\boldsymbol{\lambda}_\ell(k) \propto \frac{qe^{-\frac{\left(\mathbf{r}(\ell,k)-d_\ell\right)^2}{2\tau^2}}}{(1-q)e^{-\frac{\mathbf{r}(\ell,k)^2}{2\tau^2}}+qe^{-\frac{\left(\mathbf{r}(\ell,k)-d_\ell\right)^2}{2\tau^2}}}. \tag{70}$$

The '$\propto$' symbol accounts for the normalization constant that makes $\|\boldsymbol{\lambda}_\ell\|_1 = 1$. At low signal-to-noise ratios, this estimator remains very uninformative and it outputs near uniform probabilities. However, as the signal-to-noise ratio increases, the mass of the estimated vector starts to concentrate on fewer entries. This is the type of behavior needed for outer decoding to help improve AMP performance.

B. Global Estimates

Proceeding forward, we turn our attention to the *a posteriori* probability of message sequence $\hat{\mathbf{m}}_1,\ldots,\hat{\mathbf{m}}_{K_\mathrm{a}}$ conditioned on $\mathbf{r}$. Taking into consideration the parity requirements for valid messages through indicator function $\mathcal{G}(\cdot)$, we can write

$$\begin{aligned}&\Pr\left(\hat{\mathbf{m}}_1\ldots,\hat{\mathbf{m}}_{K_\mathrm{a}}|\mathbf{r}\right)\\ &\propto \prod_{i\in[K_\mathrm{a}]}\mathcal{G}\left(\hat{\mathbf{m}}_i\right)\prod_{j\in[L]}\Pr\left(\hat{\mathbf{m}}_1(j)\ldots,\hat{\mathbf{m}}_{K_\mathrm{a}}(j)|\mathbf{r}(j)\right).\end{aligned} \tag{71}$$

The first component in the product enforces parity consistency on every message individually. The second term comes from the Gaussian model for the effective observation. Assuming that the message components are distinct and incorporating the approximate expressions derived in the previous section, this equation evolves into

$$\begin{aligned}&\Pr\left(\hat{\mathbf{m}}_1\ldots,\hat{\mathbf{m}}_{K_\mathrm{a}}|\mathbf{r}\right)\\ &\propto \prod_{i\in[K_\mathrm{a}]}\mathcal{G}\left(\hat{\mathbf{m}}_i\right)\prod_{j\in[L]}\Pr\left(\hat{\mathbf{m}}_1(j)\ldots,\hat{\mathbf{m}}_{K_\mathrm{a}}(j)|\mathbf{r}(j)\right)\\ &\approx \left(\prod_{i\in[K_\mathrm{a}]}\mathcal{G}\left(\hat{\mathbf{m}}_i\right)\right)\left(\prod_{j\in[L]}\prod_{i\in[K_\mathrm{a}]}\Pr\left(\hat{\mathbf{m}}_i(j)|\mathbf{r}(j)\right)\right)\\ &\approx \prod_{i\in[K_\mathrm{a}]}\left(\mathcal{G}\left(\hat{\mathbf{m}}_i\right)\prod_{j\in[L]}\Pr\left(\hat{\mathbf{m}}_i(j)|\mathbf{r}(j)\right)\right).\end{aligned} \tag{72}$$

The terms in (72) decompose into a product form, one probability element for every active device. We can further simplify this expression using the marginal estimates from (70). Under the same conditions as above, the probability that a fixed device has message $\hat{\mathbf{m}}$ can be approximated by

$$\begin{aligned}\Pr\left(\hat{\mathbf{m}}|\mathbf{r}\right) &\propto \mathcal{G}\left(\hat{\mathbf{m}}\right)\prod_{\ell\in[L]}\Pr\left(\hat{\mathbf{m}}(\ell)|\mathbf{r}(\ell)\right)\\ &\approx \mathcal{G}\left(\hat{\mathbf{m}}\right)\prod_{\ell\in[L]}\boldsymbol{\lambda}_\ell(k_\ell)\end{aligned} \tag{73}$$

where k_ℓ is the location of the unique non-zero entry in $\hat{\mathbf{m}}(\ell)$. When interpreted this way and assuming the K_a messages have distinct components, $\Pr\left(\hat{\mathbf{m}}(\ell)|\mathbf{r}(\ell)\right)$ becomes bounded above by $1/K_\mathrm{a}$ because the selected device is equally likely to have sent any of the true messages. We mention briefly that it is possible to take advantage of this symmetric condition

while conducting message passing, We also see that the information afforded by the effective observation enters this equation through the local measure $\boldsymbol{\lambda}_\ell$. It then suffices to collect summary vectors $(\boldsymbol{\lambda}_\ell : \ell \in [L])$, where $\boldsymbol{\lambda}_\ell$ has length m_ℓ, to run belief propagation on the factor graph induced by the outer code.

Inspecting (73) reveals that the graphical structure of the outer code is linked to the probability that a fixed device picks a certain index for a particular block. We connect this quantity explicitly to the variable nodes on the factor graph by interpreting $p_{s_\ell}(k)$ as an estimate of the probability that this fixed device has a message $\mathbf{m}$ containing block k in section ℓ. Accordingly, we design message passing rules to deliver estimates for $p_{s_\ell}(k)$ upon completion of the process. We can write the block estimates in a format that highlights the underlying factors,

$$\begin{aligned} p_{s_\ell}(k) &\propto \sum_{\mathbf{k}:k_\ell=k} \mathcal{G}\left(\mathbf{k}\right) \prod_{\ell\in[L]} \boldsymbol{\lambda}_\ell(k_\ell) \\ &= \sum_{\mathbf{k}:k_\ell=k} \prod_{a\in\mathcal{P}} \mathcal{G}_a\left(\mathbf{k}_a\right) \prod_{j\in[L]} \boldsymbol{\lambda}_j(k_j), \end{aligned} \tag{74}$$

where $\mathbf{k} = (k_1, \ldots, k_{K_\mathrm{a}})$ is the compact index notation for $\hat{\mathbf{m}}$ obtained through the relation $k_j = [\hat{\mathbf{v}}(j)]_2$. Also, we use the localized compact notation applied to neighborhoods, $\mathbf{k}_a = (k_j : j \in N(a))$. Function $\mathcal{G}(\cdot)$ verifies the parity consistency of its vector argument under the outer code structure, as defined in (8). For the outer code, the validity characteristic function $\mathcal{G}(\cdot)$ becomes the product of local factor functions $\{\mathcal{G}_a(\cdot) : a \in \mathcal{P}\}$.

Before deriving message passing rules, we must account for the contribution of the local observations through $(\boldsymbol{\lambda}_\ell : \ell \in [L])$. Following established techniques [30], this is achieved by augmenting the graph with the (trivial) factor nodes afforded by the sections of $\mathbf{r}$. Note that these latter factors have only one connection each and, as such, their messages are static; they do not evolved with iterations. With this augmentation, we obtain standard expressions for the message passing rules. Throughout, we simply fold the static messages in the expressions without explicitly defining the augmented graph, as this step is common to the treatment of codes. The message passing rules compute estimates of the marginal distributions $(p_{s_\ell} : \ell \in [L])$.

C. Message Passing Rules

A message passed from check node a_p to variable node $s \in N(a_p)$ subscribes to the format

$$\boldsymbol{\mu}_{a_p\to s}(k) = \sum_{\mathbf{k}_{a_p}:k_p=k} \mathcal{G}_{a_p}\left(\mathbf{k}_{a_p}\right) \prod_{s_j\in N(a_p)\setminus s} \boldsymbol{\mu}_{s_j\to a_p}(k_j). \tag{75}$$

Similarly, a message going from variable node s_ℓ to check node $a \in N(s_\ell)$ assumes the form

$$\boldsymbol{\mu}_{s_\ell\to a}(k) \propto \boldsymbol{\lambda}_\ell(k) \prod_{a_p\in N(s_\ell)\setminus a} \boldsymbol{\mu}_{a_p\to s_\ell}(k). \tag{76}$$

The '$\propto$' symbol indicates that the measure is renormalized before being sent as a message. All the dynamic messages are initialized with $\boldsymbol{\mu}_{s\to a} = \mathbf{1}$ and $\boldsymbol{\mu}_{a\to s} = \mathbf{1}$. The parallel sum-product algorithm then iterates between (75) and (76). At any stage of this iterative process, the estimated marginal distribution of a specific device having transmitted block k at variable node s_ℓ is proportional to the product of the current messages from all adjoining factors,

$$p_{s_\ell}(k) \propto \boldsymbol{\lambda}_\ell(k) \prod_{a\in N(s)} \boldsymbol{\mu}_{a\to s_\ell}(k). \tag{77}$$

As usual, this procedure is guaranteed to converge for acyclic graphical models, but not for arbitrary graphs. Still, it is known to perform well in many cases where the factor graph features cycles. In our proposed algorithm, the number of belief propagation steps within one composite AMP algorithm is envisioned to be small.

Appendix C
Lipschitz Dynamic PME Denoiser

In this section, we show that the dynamic PME denoiser, with one round of message passing on the factor graph of the outer code, is Lipschitz continuous. The reader may notice that even though the treatment of this proof is restricted to triadic designs for the outer code, it can be extended to accommodate cases beyond those discussed in this paper. Our proof technique may serve as a blueprint to analyze generic architectures which use a combination of inner AMP and an outer decoder with dynamic interactions between the two. Our strategy is to show that the magnitudes of the entries in the Jacobian matrix of $\boldsymbol{\eta}_t^{\mathrm{PME}}(\mathbf{r})$ with respect to $\mathbf{r}$ are uniformly bounded. Recall that

$$\begin{aligned} \boldsymbol{\eta}_t^{\mathrm{PME}}(\mathbf{r}) &= \hat{\mathbf{s}}_1^{\mathrm{PME}}\left(\mathbf{r}, \tau_t\right) \cdots \hat{\mathbf{s}}_L^{\mathrm{PME}}\left(\mathbf{r}, \tau_t\right) \\ \hat{\mathbf{s}}_\ell^{\mathrm{PME}}\left(\mathbf{r}, \tau_t\right) &= \big(\hat{s}_\ell\left(\mathbf{q}(\ell,k), \mathbf{r}(\ell,k), \tau_t\right) : k \in 0, \ldots, m_\ell - 1\big). \end{aligned}$$

Consequently, we are interested in objects of the form

$$\frac{\partial \hat{s}_\ell\left(\mathbf{q}(\ell,k), \mathbf{r}(\ell,k), \tau_t\right)}{\partial \mathbf{r}(j,k')}.$$

There are three categories to consider: $j = \ell$, $j \in N(s_\ell)$, and $j \notin \{\ell\} \cup N(s_\ell)$. The third group is the easiest to treat because the partial derivatives are each equal to zero when only one round of belief propagation is performed on the underlying factor graph. Next, we turn our attention to cases where $j = \ell$. Applying Lemma 4, we immediately get

$$\begin{aligned} &\left| \frac{\partial \hat{s}_\ell\left(\mathbf{q}(\ell,k), \mathbf{r}(\ell,k), \tau_t\right)}{\partial \mathbf{r}(\ell,k)} \right| \\ &= \frac{d_\ell}{\tau^2} \hat{s}_\ell\left(\mathbf{q}(\ell,k), \mathbf{r}(\ell,k), \tau_t\right) \left(1 - \hat{s}_\ell\left(\mathbf{q}(\ell,k), \mathbf{r}(\ell,k), \tau_t\right)\right) \\ &\leq \frac{d_{\max}}{2\tau^2}. \end{aligned} \tag{78}$$

Furthermore, when $k' \neq k$, we have

$$\frac{\partial \hat{s}_\ell\left(\mathbf{q}(\ell,k), \mathbf{r}(\ell,k), \tau_t\right)}{\partial \mathbf{r}(\ell,k')} = 0. \tag{79}$$

Therefore, entries in the Jacobian matrix corresponding to this category are uniformly bounded. It remains to derive a similar bound for scenarios where $j \in N(s_\ell)$. Establishing the desired

property for this category requires a few steps. We begin with an analog to Lemma 4.

Lemma 15. *The partial derivative of the PME defined in Lemma 3 with respect to q is*

$$\frac{\partial \hat{s}_\ell(q,r,\tau)}{\partial q} = \frac{\hat{s}_\ell(q,r,\tau)\left(1-\hat{s}_\ell(q,r,\tau)\right)}{q(1-q)}. \tag{80}$$

Proof: Recall that the PME can be rewritten as

$$\hat{s}_\ell(q,r,\tau) = \frac{q}{q+(1-q)\exp\left(\frac{d_\ell^2-2rd_\ell}{2\tau^2}\right)}. \tag{81}$$

By the chain rule of differentiation, we get

$$\begin{aligned}\frac{\partial \hat{s}_\ell(q,r,\tau)}{\partial q} &= \frac{\exp\left(\frac{d_\ell^2-2rd_\ell}{2\tau^2}\right)}{\left(q+(1-q)\exp\left(\frac{d_\ell^2-2rd_\ell}{2\tau^2}\right)\right)^2} \\ &= \frac{\hat{s}_\ell(q,r,\tau)\left(1-\hat{s}_\ell(q,r,\tau)\right)}{q(1-q)},\end{aligned}$$

as stated. ∎

Our next step is to link $\mathbf{q}(\ell,k)$ to quantities defined on the factor graph of the outer code. Recall that we introduced $\mathbf{q}(\ell,k)$ in (39) with

$$\mathbf{q}(\ell,k) = 1-\left(1-\frac{\boldsymbol{\mu}_{s_\ell}(k)}{\|\boldsymbol{\mu}_{s_\ell}\|_1}\right)^{K_\mathrm{a}}. \tag{82}$$

As an immediate consequence of this definition, we obtain the inequality

$$\begin{aligned}\mathbf{q}(\ell,k) &= 1-\left(1-\frac{\boldsymbol{\mu}_{s_\ell}(k)}{\left\|\boldsymbol{\mu}_{s_\ell}\right\|_1}\right)^{K_\mathrm{a}} \\ &\geq 1-\left(1-\frac{\boldsymbol{\mu}_{s_\ell}(k)}{\left\|\boldsymbol{\mu}_{s_\ell}\right\|_1}\right) = \frac{\boldsymbol{\mu}_{s_\ell}(k)}{\left\|\boldsymbol{\mu}_{s_\ell}\right\|_1}.\end{aligned} \tag{83}$$

Another straightforward, yet useful result related to $\mathbf{q}(\ell,k)$ is

$$\begin{aligned}\frac{\partial \mathbf{q}(\ell,k)}{\partial \mathbf{r}(j,k')} &= K_\mathrm{a}\left(1-\frac{\boldsymbol{\mu}_{s_\ell}(k)}{\|\boldsymbol{\mu}_{s_\ell}\|_1}\right)^{K_\mathrm{a}-1}\frac{\partial}{\partial \mathbf{r}(j,k')}\frac{\boldsymbol{\mu}_{s_\ell}(k)}{\left\|\boldsymbol{\mu}_{s_\ell}\right\|_1} \\ &= K_\mathrm{a}\frac{1-\mathbf{q}(\ell,k)}{1-\frac{\boldsymbol{\mu}_{s_\ell}(k)}{\|\boldsymbol{\mu}_{s_\ell}\|_1}}\frac{\partial}{\partial \mathbf{r}(j,k')}\frac{\boldsymbol{\mu}_{s_\ell}(k)}{\left\|\boldsymbol{\mu}_{s_\ell}\right\|_1}.\end{aligned} \tag{84}$$

Then, by Lemma 15 and (84), we have

$$\begin{aligned}&\frac{\partial \hat{s}_\ell\left(\mathbf{q}(\ell,k),\mathbf{r}(\ell,k),\tau\right)}{\partial \mathbf{r}(j,k')} \\ &= \frac{\hat{s}_\ell(\mathbf{q}(\ell,k),\mathbf{r}(\ell,k),\tau)\left(1-\hat{s}_\ell(\mathbf{q}(\ell,k),\mathbf{r}(\ell,k),\tau)\right)}{\mathbf{q}(\ell,k)\left(1-\mathbf{q}(\ell,k)\right)}\frac{\partial \mathbf{q}(\ell,k)}{\partial \mathbf{r}(j,k')} \\ &= K_\mathrm{a}\frac{\hat{s}_\ell(\mathbf{q}(\ell,k),\mathbf{r}(\ell,k),\tau)\left(1-\hat{s}_\ell(\mathbf{q}(\ell,k),\mathbf{r}(\ell,k),\tau)\right)}{\mathbf{q}(\ell,k)\left(1-\frac{\boldsymbol{\mu}_{s_\ell}(k)}{\|\boldsymbol{\mu}_{s_\ell}\|_1}\right)}\times \\ &\quad \frac{\partial}{\partial \mathbf{r}(j,k')}\frac{\boldsymbol{\mu}_{s_\ell}(k)}{\left\|\boldsymbol{\mu}_{s_\ell}\right\|_1}.\end{aligned} \tag{85}$$

Taking the absolute value of the gradient and incorporating (83), we get

$$\begin{aligned}&\left|\frac{\partial \hat{s}_\ell\left(\mathbf{q}(\ell,k),\mathbf{r}(\ell,k),\tau\right)}{\partial \mathbf{r}(j,k')}\right| \\ &= K_\mathrm{a}\frac{\hat{s}_\ell(\mathbf{q}(\ell,k),\mathbf{r}(\ell,k),\tau)\left(1-\hat{s}_\ell(\mathbf{q}(\ell,k),\mathbf{r}(\ell,k),\tau)\right)}{\mathbf{q}(\ell,k)\left(1-\frac{\boldsymbol{\mu}_{s_\ell}(k)}{\|\boldsymbol{\mu}_{s_\ell}\|_1}\right)}\times \\ &\quad \left|\frac{\partial}{\partial \mathbf{r}(j,k')}\frac{\boldsymbol{\mu}_{s_\ell}(k)}{\left\|\boldsymbol{\mu}_{s_\ell}\right\|_1}\right| \\ &\leq \frac{K_\mathrm{a}}{2}\frac{1}{\frac{\boldsymbol{\mu}_{s_\ell}(k)}{\|\boldsymbol{\mu}_{s_\ell}\|_1}\left(1-\frac{\boldsymbol{\mu}_{s_\ell}(k)}{\|\boldsymbol{\mu}_{s_\ell}\|_1}\right)}\left|\frac{\partial}{\partial \mathbf{r}(j,k')}\frac{\boldsymbol{\mu}_{s_\ell}(k)}{\left\|\boldsymbol{\mu}_{s_\ell}\right\|_1}\right|.\end{aligned} \tag{86}$$

At this point, we shift our focus to the partial derivative of $\boldsymbol{\mu}_{s_\ell}(k)/\left\|\boldsymbol{\mu}_{s_\ell}\right\|_1$. For the purpose of this derivation, we introduce the compact notation

$$\boldsymbol{\mu}_{s_\ell\sim a}(k) = \prod_{a_p\in N(s_\ell)\setminus a}\boldsymbol{\mu}_{a_p\to s_\ell}(k),$$

where $a\in N(s_\ell)$. It is pertinent to note that, in triadic designs, $N(s_\ell)\setminus a$ is the empty set whenever s_ℓ is a parity section. This is because, in such cases, the cardinality of $N(s_\ell)$ is one, as illustrated in Fig. 7. For such nodes, we take $\boldsymbol{\mu}_{s_\ell\sim a}(k)=1$ for all values of k.

Getting back to the proof of the last category, let a be the sole check in $N(s_\ell)$ such that $a\in N(s_j)$. In other words, $a\in N(s_\ell)\cap N(s_j)$ denotes the unique check node shared by s_ℓ and s_j in the triadic design. Then, we can write

$$\begin{aligned}\frac{\partial}{\partial \mathbf{r}(j,k')}\boldsymbol{\mu}_{s_\ell}(k) &= \frac{\partial}{\partial \mathbf{r}(j,k')}\prod_{a_p\in N(s_\ell)}\boldsymbol{\mu}_{a_p\to s_\ell}(k) \\ &= \boldsymbol{\mu}_{s_\ell\sim a}(k)\frac{\partial}{\partial \mathbf{r}(j,k')}\sum_{k_1+k_2\equiv k}\boldsymbol{\lambda}_j^{\mathrm{PME}}(k_1)\boldsymbol{\lambda}_{j^*}^{\mathrm{PME}}(k_2) \\ &= \boldsymbol{\mu}_{s_\ell\sim a}(k)\frac{\partial}{\partial \mathbf{r}(j,k')}\boldsymbol{\lambda}_j^{\mathrm{PME}}(k')\boldsymbol{\lambda}_{j^*}^{\mathrm{PME}}(k-k') \\ &= \frac{d_j}{\tau^2}\boldsymbol{\mu}_{s_\ell\sim a}(k)\left(1-\boldsymbol{\lambda}_j^{\mathrm{PME}}(k')\right)\boldsymbol{\lambda}_j^{\mathrm{PME}}(k')\boldsymbol{\lambda}_{j^*}^{\mathrm{PME}}(k-k')\end{aligned} \tag{87}$$

where s_{j^*} is the unique section that forms a triad with s_ℓ and s_j through check node a, i.e., $N(a)=\{s_\ell,s_j,s_{j^*}\}$. Likewise, we can compute the partial derivative of $\boldsymbol{\mu}_{s_\ell}(k)/\left\|\boldsymbol{\mu}_{s_\ell}\right\|_1$. This derivative appears in (88), where we have leveraged (87) to simplify the expression. From (86) and (88), we get the inequality in (89). We can upper bound the last term in this

$$\begin{aligned}\frac{\partial}{\partial \mathbf{r}(j,k')}\frac{\boldsymbol{\mu}_{s_\ell}(k)}{\left\|\boldsymbol{\mu}_{s_\ell}\right\|_1} &= \frac{1}{\left\|\boldsymbol{\mu}_{s_\ell}\right\|_1}\frac{\partial \boldsymbol{\mu}_{s_\ell}(k)}{\partial \mathbf{r}(j,k')} - \frac{\boldsymbol{\mu}_{s_\ell}(k)}{\left\|\boldsymbol{\mu}_{s_\ell}\right\|_1^2}\sum_{k_0}\frac{\partial \boldsymbol{\mu}_{s_\ell}(k_0)}{\partial \mathbf{r}(j,k')} \\ &= \frac{d_j}{\tau^2}\left(1-\boldsymbol{\lambda}_j^{\mathrm{PME}}(k')\right)\frac{\boldsymbol{\lambda}_j^{\mathrm{PME}}(k')}{\left\|\boldsymbol{\mu}_{s_\ell}\right\|_1}\left(\boldsymbol{\mu}_{s_\ell\sim a}(k)\boldsymbol{\lambda}_{j^*}^{\mathrm{PME}}(k-k') - \frac{\boldsymbol{\mu}_{s_\ell}(k)}{\left\|\boldsymbol{\mu}_{s_\ell}\right\|_1}\sum_{k_0}\boldsymbol{\mu}_{s_\ell\sim a}(k_0)\boldsymbol{\lambda}_{j^*}^{\mathrm{PME}}(k_0-k')\right)\end{aligned} \tag{88}$$

$$\begin{aligned}&\left|\frac{\partial \hat{s}_\ell\left(\mathbf{q}(\ell,k),\mathbf{r}(\ell,k),\tau\right)}{\partial \mathbf{r}(j,k')}\right| \\ &\leq \frac{K_{\mathrm{a}}}{2}\frac{d_j}{\tau^2}\frac{\left(1-\boldsymbol{\lambda}_j^{\mathrm{PME}}(k')\right)}{\frac{\boldsymbol{\mu}_{s_\ell}(k)}{\|\boldsymbol{\mu}_{s_\ell}\|_1}\left(1-\frac{\boldsymbol{\mu}_{s_\ell}(k)}{\|\boldsymbol{\mu}_{s_\ell}\|_1}\right)}\left|\frac{\boldsymbol{\lambda}_j^{\mathrm{PME}}(k')\boldsymbol{\lambda}_{j^*}^{\mathrm{PME}}(k-k')\boldsymbol{\mu}_{s_\ell\sim a}(k)}{\left\|\boldsymbol{\mu}_{s_\ell}\right\|_1} - \frac{\boldsymbol{\mu}_{s_\ell}(k)}{\left\|\boldsymbol{\mu}_{s_\ell}\right\|_1}\sum_{k_0}\frac{\boldsymbol{\lambda}_j^{\mathrm{PME}}(k')\boldsymbol{\lambda}_{j^*}^{\mathrm{PME}}(k_0-k')\boldsymbol{\mu}_{s_\ell\sim a}(k_0)}{\left\|\boldsymbol{\mu}_{s_\ell}\right\|_1}\right|.\end{aligned} \tag{89}$$

inequality as follows,

$$\begin{aligned}&\left|\frac{\boldsymbol{\lambda}_j^{\mathrm{PME}}(k')\boldsymbol{\lambda}_{j^*}^{\mathrm{PME}}(k-k')\boldsymbol{\mu}_{s_\ell\sim a}(k)}{\left\|\boldsymbol{\mu}_{s_\ell}\right\|_1}\right. \\ &\quad\left. -\frac{\boldsymbol{\mu}_{s_\ell}(k)}{\left\|\boldsymbol{\mu}_{s_\ell}\right\|_1}\sum_{k_0}\frac{\boldsymbol{\lambda}_j^{\mathrm{PME}}(k')\boldsymbol{\lambda}_{j^*}^{\mathrm{PME}}(k_0-k')\boldsymbol{\mu}_{s_\ell\sim a}(k_0)}{\left\|\boldsymbol{\mu}_{s_\ell}\right\|_1}\right| \\ &= \left|\left(1-\frac{\boldsymbol{\mu}_{s_\ell}(k)}{\left\|\boldsymbol{\mu}_{s_\ell}\right\|_1}\right)\frac{\boldsymbol{\lambda}_j^{\mathrm{PME}}(k')\boldsymbol{\lambda}_{j^*}^{\mathrm{PME}}(k-k')\boldsymbol{\mu}_{s_\ell\sim a}(k)}{\left\|\boldsymbol{\mu}_{s_\ell}\right\|_1}\right. \\ &\quad\left. -\frac{\boldsymbol{\mu}_{s_\ell}(k)}{\left\|\boldsymbol{\mu}_{s_\ell}\right\|_1}\sum_{k_0\neq k}\frac{\boldsymbol{\lambda}_j^{\mathrm{PME}}(k')\boldsymbol{\lambda}_{j^*}^{\mathrm{PME}}(k_0-k')\boldsymbol{\mu}_{s_\ell\sim a}(k_0)}{\left\|\boldsymbol{\mu}_{s_\ell}\right\|_1}\right| \\ &\leq \left|\left(1-\frac{\boldsymbol{\mu}_{s_\ell}(k)}{\left\|\boldsymbol{\mu}_{s_\ell}\right\|_1}\right)\frac{\boldsymbol{\lambda}_j^{\mathrm{PME}}(k')\boldsymbol{\lambda}_{j^*}^{\mathrm{PME}}(k-k')\boldsymbol{\mu}_{s_\ell\sim a}(k)}{\left\|\boldsymbol{\mu}_{s_\ell}\right\|_1}\right| \\ &\quad + \left|\frac{\boldsymbol{\mu}_{s_\ell}(k)}{\left\|\boldsymbol{\mu}_{s_\ell}\right\|_1}\sum_{k_0\neq k}\frac{\boldsymbol{\lambda}_j^{\mathrm{PME}}(k')\boldsymbol{\lambda}_{j^*}^{\mathrm{PME}}(k_0-k')\boldsymbol{\mu}_{s_\ell\sim a}(k_0)}{\left\|\boldsymbol{\mu}_{s_\ell}\right\|_1}\right| \\ &\leq \left|\left(1-\frac{\boldsymbol{\mu}_{s_\ell}(k)}{\left\|\boldsymbol{\mu}_{s_\ell}\right\|_1}\right)\frac{\boldsymbol{\mu}_{s_\ell}(k)}{\left\|\boldsymbol{\mu}_{s_\ell}\right\|_1}\right| + \left|\frac{\boldsymbol{\mu}_{s_\ell}(k)}{\left\|\boldsymbol{\mu}_{s_\ell}\right\|_1}\left(1-\frac{\boldsymbol{\mu}_{s_\ell}(k)}{\left\|\boldsymbol{\mu}_{s_\ell}\right\|_1}\right)\right| \\ &= 2\frac{\boldsymbol{\mu}_{s_\ell}(k)}{\left\|\boldsymbol{\mu}_{s_\ell}\right\|_1}\left(1-\frac{\boldsymbol{\mu}_{s_\ell}(k)}{\left\|\boldsymbol{\mu}_{s_\ell}\right\|_1}\right).\end{aligned} \tag{90}$$

Combining (89), (90), and the fact that $1-\boldsymbol{\lambda}_{j_1}^{\mathrm{PME}}(k') \leq 1$, we arrive at a uniform bound for the last category,

$$\left|\frac{\partial \hat{s}_\ell\left(\mathbf{q}(\ell,k),\mathbf{r}(\ell,k),\tau\right)}{\partial \mathbf{r}(j,k')}\right| \leq \frac{K_{\mathrm{a}} d_{\max}}{\tau^2}. \tag{91}$$

Altogether, we gather that the components of the Jacobian matrix are uniformly bounded and, consequently, we conclude that the denoiser is Lipschitz continuous.

REFERENCES

[1] Yury Polyanskiy, "A perspective on massive random-access," in *Proc. Int. Symp. Inf. Theory*, 2017, pp. 2523–2527.

[2] Or Ordentlich and Yury Polyanskiy, "Low complexity schemes for the random access Gaussian channel," in *Proc. Int. Symp. Inf. Theory*, 2017, pp. 2528–2532.

[3] Avinash Vem, Krishna R. Narayanan, Jean-Francois Chamberland, and Jun Cheng, "A user-independent successive interference cancellation based coding scheme for the unsourced random access Gaussian channel," *IEEE Trans. Commun.*, vol. 67, no. 12, pp. 8258–8272, 2019.

[4] Vamsi K. Amalladinne, Jean-Francois Chamberland, and Krishna R. Narayanan, "A coded compressed sensing scheme for unsourced multiple access," *IEEE Trans. Inf. Theory*, vol. 66, no. 10, pp. 6509–6533, October 2020.

[5] Alexander Fengler, Peter Jung, and Giuseppe Caire, "SPARCs for unsourced random access," *IEEE Trans. Inf. Theory*, vol. 67, no. 10, pp. 6894–6915, October 2021.

[6] Asit Pradhan, Vamsi Amalladinne, Avinash Vem, Krishna R. Narayanan, and Jean-Francois Chamberland, "A joint graph based coding scheme for the unsourced random access Gaussian channel," in *Proc. Global Telecommun. Conf.* IEEE, 2019.

[7] Robert Calderbank and Andrew Thompson, "CHIRRUP: A practical algorithm for unsourced multiple access," *Information and Inference*, vol. 9, no. 4, pp. 875–897, 2019.

[8] Vamsi K. Amalladinne, Jean-Francois Chamberland, and Krishna R. Narayanan, "An enhanced decoding algorithm for coded compressed sensing," in *Proc. Int. Conf. Acoust., Speech, Signal Process.* IEEE, 2020.

[9] Evgeny Marshakov, Gleb Balitskiy, Kirill Andreev, and Alexey Frolov, "A polar code based unsourced random access for the Gaussian MAC," in *Proc. Vehicular Technol. Conf.* IEEE, 2019.

[10] Asit K. Pradhan, Vamsi K. Amalladinne, Krishna R. Narayanan, and Jean-Francois Chamberland, "Polar coding and random spreading for unsourced multiple access," in *Proc. Int. Conf. Commun.* IEEE, 2020.

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

[12] Andrea Montanari, "Graphical models concepts in compressed sensing," in *Compressed Sensing: Theory and Applications*, Yonina C. Eldar and Gitta Kutyniok, Eds., chapter 9. Cambridge, 2012.

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

[14] Ramji Venkataramanan, Sekhar Tatikonda, and Andrew Barron, "Sparse regression codes," *Foundations and Trends in Communications and Information Theory*, vol. 15, no. 1-2, pp. 1–195, 2019.

[15] Cynthia Rush, Adam Greig, and Ramji Venkataramanan, "Capacity-achieving sparse superposition codes via approximate message passing decoding," *IEEE Trans. Inf. Theory*, vol. 63, no. 3, pp. 1476–1500, 2017.

[16] Jean Barbier and Florent Krzakala, "Approximate message-passing decoder and capacity achieving sparse superposition codes," *IEEE Trans. Inf. Theory*, vol. 63, no. 8, pp. 4894–4927, 2017.

[17] Jean Barbier and Florent Krzakala, "Replica analysis and approximate message passing decoder for superposition codes," in *Proc. Int. Symp. Inf. Theory*. IEEE, 2014, pp. 1494–1498.

[18] Antony Joseph and Andrew R. Barron, "Fast sparse superposition codes have near exponential error probability for $R < C$," *IEEE Trans. Inf. Theory*, vol. 60, no. 2, pp. 919–942, 2013.

[19] Antony Joseph and Andrew R. Barron, "Least squares superposition codes of moderate dictionary size are reliable at rates up to capacity," *IEEE Trans. Inf. Theory*, vol. 58, no. 5, pp. 2541–2557, 2012.
[20] Adam Greig and Ramji Venkataramanan, "Techniques for improving the finite length performance of sparse superposition codes," *IEEE Trans. Commun.*, vol. 66, no. 3, pp. 905–917, 2017.
[21] Jean Barbier, Christophe Schülke, and Florent Krzakala, "Approximate message-passing with spatially coupled structured operators, with applications to compressed sensing and sparse superposition codes," *Journal of Statistical Mechanics: Theory and Experiment*, vol. 2015, no. 5, pp. P05013, 2015.
[22] Cynthia Rush, Kuan Hsieh, and Ramji Venkataramanan, "Capacity-achieving spatially coupled sparse superposition codes with AMP decoding," *IEEE Trans. Inf. Theory*, vol. 67, no. 7, pp. 4446–4484, 2021.
[23] Kuan Hsieh and Ramji Venkataramanan, "Modulated sparse superposition codes for the complex AWGN channel," *IEEE Trans. Inf. Theory*, vol. 67, no. 7, pp. 4385–4404, 2021.
[24] Cynthia Rush and Ramji Venkataramanan, "The error probability of sparse superposition codes with approximate message passing decoding," *IEEE Trans. Inf. Theory*, vol. 65, no. 5, pp. 3278–3303, 2018.
[25] Cynthia Rush, Kuan Hsieh, and Ramji Venkataramanan, "Spatially coupled sparse regression codes with sliding window AMP decoding," in *Proc. Inf. Theory Workshop*. IEEE, 2019.
[26] Kuan Hsieh, Cynthia Rush, and Ramji Venkataramanan, "Spatially coupled sparse regression codes: Design and state evolution analysis," in *Proc. Int. Symp. Inf. Theory*. IEEE, 2018, pp. 1016–1020.
[27] Shansuo Liang, Chulong Liang, Junjie Ma, and Li Ping, "Compressed coding, AMP based-decoding, and analog spatial coupling," *IEEE Trans. Commun.*, vol. 68, no. 12, pp. 7362–7375, 2020.
[28] Vamsi K Amalladinne, Krishna R Narayanan, Jean-Francois Chamberland, and Dongning Guo, "Asynchronous neighbor discovery using coupled compressive sensing," in *Proc. Int. Conf. Acoust., Speech, Signal Process.* IEEE, 2019, pp. 4569–4573.
[29] Vamsi K. Amalladinne, Avinash Vem, Dileep Kumar Soma, Krishna R. Narayanan, and Jean-Francois Chamberland, "A coupled compressive sensing scheme for unsourced multiple access," in *Proc. Int. Conf. Acoust., Speech, Signal Process.* IEEE, 2018, pp. 6628–6632.
[30] Frank R. Kschischang, Brendan J Frey, and Hans-Andrea Loeliger, "Factor graphs and the sum-product algorithm," *IEEE Trans. Inf. Theory*, vol. 47, no. 2, pp. 498–519, 2001.
[31] Mohsen Bayati and Andrea 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, 2011.
[32] David L. Donoho, Adel Javanmard, and Andrea Montanari, "Information-theoretically optimal compressed sensing via spatial coupling and approximate message passing," *IEEE Trans. Inf. Theory*, vol. 59, no. 11, pp. 7434–7464, 2013.
[33] Raphael Berthier, Andrea Montanari, and Phan-Minh Nguyen, "State evolution for approximate message passing with non-separable functions," *Information and Inference*, vol. 9, no. 1, pp. 33–79, 2020.
[34] Philip Schniter, "A simple derivation of AMP and its state evolution via first-order cancellation," *IEEE Trans. Signal Process.*, vol. 68, pp. 4283–4292, 2020.

Vamsi K. Amalladinne received the B.Tech. degree in Electronics and Communication Engineering from the LNM Institute of Information Technology, Jaipur, India, in 2012, the M.Tech. degree in Signal Processing and Communications from the Indian Institute of Technology Kanpur, India, in 2014, and the Ph.D. degree in Electrical and Computer Engineering at Texas A&M University, College Station, USA in 2021. From June 2014 to August 2016, he was employed as a DSP Firmware developer for CDMA systems at Qualcomm, Hyderabad, India. He is currently employed as a senior research engineer at the wireless research & development division of Qualcomm, San Diego, USA. His research interests are in wireless communication, signal processing, error control coding and compressed sensing.

Asit Kumar Pradhan received the B.Tech. degree from Biju Patnaik University of Technology, Bhubaneswar, India, in 2010, and the M.Tech. and Ph.D. degrees in electrical engineering from the Indian Institute of Technology Madras, Chennai, India, in 2013 and 2018, respectively. He was a postdoctoral fellow at Texas A&M University from 2018 to 2021. He is currently a postdoctoral researcher at University of Arizona. His research interests include information theory and coding theory.

Cynthia Rush (Member, IEEE) received the B.S. degree in mathematics from the University of North Carolina at Chapel Hill in 2010, and the M.A. and Ph.D. degrees in statistics from Yale University in 2011 and 2016, respectively. She is currently the Howard Levene Assistant Professor of statistics with the Department of Statistics, Columbia University. Her research interests include high-dimensional statistics, information theory, and the mathematical foundations of machine learning.

Jean-Francois Chamberland (S'98–M'04–SM'09) received the Ph.D. degree from the University of Illinois at Urbana-Champaign. He is currently a Professor with the Department of Electrical and Computer Engineering at Texas A&M University. His research interests are in the areas of computing, information, and inference. He has been a recipient of the IEEE Young Author Best Paper Award from the IEEE Signal Processing Society and the Faculty Early Career Development (CAREER) Award from the National Science Foundation. He served as an Associate Editor for the IEEE Transactions on Information Theory from 2017 to 2020,

Krishna R. Narayanan (S'92–M'98–SM'09–F'15) received the B.E. degree from Coimbatore Institute of Technology, Coimbatore, India, the M.S. degree from Iowa State University, Ames, IA, USA, and the Ph.D. degree from Georgia Institute of Technology, Atlanta, GA, USA, in 1992, 1994, and 1998, respectively. He is currently the Eric D. Rubin '06 Professor of Electrical and Computer Engineering at Texas A&M University, College Station, TX, USA. His research interests include coding theory, information theory, signal processing with applications to wireless communications, data storage, and data science. He served as an Associate Editor for the IEEE Transactions on Information Theory from 2015 to 2018, an Area Editor for the coding theory and applications area of the IEEE Transactions on communications from 2007 to 2011. He also served on the board of governors of the IEEE Information Theory Society.