# A Coded Compressed Sensing Scheme for Unsourced Multiple Access

Vamsi K. Amalladinne, *Graduate Student Member, IEEE*, Jean-Francois Chamberland, *Senior Member, IEEE*, and Krishna R. Narayanan, *Fellow, IEEE*

*Abstract*—This article introduces a novel scheme, termed coded compressed sensing, for unsourced multiple-access communication. The proposed divide-and-conquer approach leverages recent advances in compressed sensing and forward error correction to produce a novel uncoordinated access paradigm, along with a computationally efficient decoding algorithm. Within this framework, every active device partitions its data into several sub-blocks and, subsequently, adds redundancy using a systematic linear block code. Compressed sensing techniques are then employed to recover sub-blocks up to a permutation of their order, and the original messages are obtained by stitching fragments together using a tree-based algorithm. The error probability and computational complexity of this access paradigm are characterized. An optimization framework, which exploits the tradeoff between performance and computational complexity, is developed to assign parity-check bits to each sub-block. In addition, two emblematic parity bit allocation strategies are examined and their performances are analyzed in the limit as the number of active users and their corresponding payloads tend to infinity. The number of channel uses needed and the computational complexity associated with these allocation strategies are established for various scaling regimes. Numerical results demonstrate that coded compressed sensing outperforms other existing practical access strategies over a range of operational scenarios.

*Index Terms*—Communication, forward error correction, unsourced multiple-access, compressed sensing.

## I. INTRODUCTION

UNCOORDINATED and unsourced multiple access communication (MAC) is a novel formulation for non-orthogonal multiple access. This framework, which was introduced by Polyanskiy in [1], is particularly relevant in the context of the Internet of Things (IoT). It is closely related to coded random access [2] and the many access channel [3]. In this new paradigm, a wireless network is composed of $K_{\mathrm{tot}}$ users, out of which a smaller group of $K_{\mathrm{a}}$ users are active at

any given time. These $K_{\mathrm{a}}$ active users each wish to transmit a $B$-bit message to the access point in an uncoordinated fashion. The access point is tasked with recovering the set of transmitted messages, without regard for the identities of the corresponding sources. The total number of users $K_{\mathrm{tot}}$, can be very large, whereas parameters $K_{\mathrm{a}}$ and $B$ are envisioned to be orders of magnitude smaller than $K_{\mathrm{tot}}$.

For typical IoT applications, the message length $B$ is envisioned to remain small and, in this regime, asymptotic information-theoretic results only offer limited insight. Rather, finite-length bounds are often more meaningful. Along these lines, Polyanskiy [1] offers finite block-length (FBL) achievability bounds based on random Gaussian codebooks and maximum-likelihood (ML) decoding. While this work provides a benchmark to evaluate coding schemes, random coding with ML decoding is computationally infeasible in most practical situations and, hence, there is a pressing need for computationally-efficient coding and decoding schemes aimed at the unsourced MAC.

In [4], Ordentlich and Polyanskiy show that many existing multiple access strategies, including treating inference as noise (TIN) and ALOHA perform poorly in this context, especially when $K_{\mathrm{a}}$ exceeds 100. They also propose the first low-complexity coding scheme tailored to this setting. In their scheme, the transmission period is divided into sub-blocks, or slots, and the system operates in a synchronous fashion. Specifically, all the users are aware of slot boundaries. Within this framework, every active user transmits a codeword during a randomly chosen slot. A data block is formed with a concatenated code that is designed for a $T$-user, real-addition Gaussian multiple access channel ($T$-GMAC); values for $T$ range from 2 to 5. Although this scheme performs significantly better than ALOHA and TIN, there remains an important gap of approximately 20 dB between this realized performance and the aforementioned achievability limit associated with the unsourced MAC [1]. In subsequent work [5], Vem et al. describe a low-complexity coding scheme that relies on a similar slotted structure. This latter framework consists of an improved, close-to-optimal coding strategy for the $T$-GMAC; coupled to the application of successive interference cancellation across slots to reduce the performance degradation caused by overcrowded slots. The combination of these two features constitutes a significant improvement over [4], with a performance curve that lies only approximately 6 dB away from the above mentioned achievability benchmark. Both

schemes discussed above adopt a channel coding viewpoint, wherein the $K_a$-user GMAC is reduced to multiple smaller $T$-GMAC channel problems.

In contrast, this article promotes an alternative compressed sensing (CS) approach tailored to the uncoordinated and unsourced MAC problem. Enabling uncoordinated multiple access to a massive number of users has a strong connection to the problem of support recovery in noisy compressed sensing [1]. Conceptually, decoding an instance of the uncoordinated MAC entails finding the support of an unknown vector of length $2^B$. A naive CS solution requires operations on sensing matrices with $2^B$ columns, which is computationally intractable for values of $B$ on the order of 100. Consequently, any pragmatic solution to this challenge should have a computational cost that is sub-linear in the dimension of the problem.

To this end, we devise a novel CS algorithm, called coded compressed sensing (CCS), for the unsourced MAC problem. This algorithm achieves sub-linear complexity in the dimension of the CS problem using a divide-and-conquer approach. Information messages at the users are partitioned into smaller sub-blocks such that each sub-problem is amenable to CS recovery. Before transmission, redundancy is added to individual fragments using a systematic linear code. The collection of sub-blocks transmitted within a slot are recovered using a standard CS algorithm. Once this is achieved, the individual fragments of the original messages need to be pieced together. This is accomplished via a low-complexity tree-based algorithm. The overall structure of this communication architecture yields better performance compared to previously proposed algorithms with comparable computational complexity. The main contributions of this article are summarized below.

1) A novel low-complexity compressed sensing algorithm is proposed to solve the unsourced, uncoordinated multiple access problem.
2) Explicit closed-form expressions are provided to characterize the error probability and the average computational complexity of this scheme.
3) We show how the parameters of this algorithm can be tuned to gracefully trade off complexity and performance. An optimization framework is developed to exploit this tradeoff.
4) In the finite block length regime, the algorithm is shown to perform close to the FBL achievability bounds.
5) In the asymptotic regime where $K_a$ and $B$ approach infinity, bounds are provided for the average computational complexity and the number of channel uses needed for this scheme to be asymptotically reliable. The algorithm has an average complexity that is sub-linear in the dimension of the problem.

In the section below, we review the connection between compressed sensing and multiple access, and we survey pertinent recent developments in the field of compressed sensing.

### A. Compressed Sensing and Multiple Access Schemes

The connection between compressed sensing and multiple access has been explored in the literature [6]–[9]. The work in [6] provides necessary and sufficient conditions for exact support recovery by interpreting the sparse recovery problem as a Gaussian MAC coding problem. In [7], the LASSO algorithm is employed for user identification in a random multiple access scenario. Yet, the complexity of this algorithm does not scale well in many regimes of interest. Another closely related area where compressed sensing techniques are applied for random access is the problem of discovering the access points within the range of a wireless device in a network, also known as the neighbor discovery problem [8]. This setting is characterized by devices attempting to identify the network interface addresses of nodes within a single hop. Therein, Zhang et al. propose two compressed sensing schemes based on group testing and second-order Reed Muller codes followed by chirp decoding. The second scheme is shown to have sub-linear computational cost. In [10], a low complexity neighbor discovery scheme, referred to as sparse-orthogonal frequency division multiplexing (sparse-OFDM), which is based on the recent developments in sparse Fourier transform [11], is proposed for the asynchronous neighbor discovery problem. Still, there does not appear to exist a straightforward way to extrapolate these schemes to the unsourced MAC problem and, hence, their connection to the FBL bounds for this latter problem remains unclear. In a recent contribution [12], we make a performance comparison between CCS and the sparse-OFDM scheme found in [10]; we demonstrate that CCS significantly outperforms sparse-OFDM scheme for asynchronous neighbor discovery.

Schemes that combine random access and compressed sensing have been proposed in [13] and a survey is presented in [9]. A multiple access scheme that uses compressed sensing in sensor networks is presented in [14]. Still, none of these articles use a divide-and-conquer strategy or a tree-like coded structure. As such, our proposed coding scheme forms a substantial departure from previous contributions in the field of communication. Recently, we became aware that the structure of CCS shares similarities with divide-and-conquer approaches that have appeared in theoretical computer science, see e.g., [15], [16]. Therein, the authors also advocate the use of an outer code before fragmentation, followed by the application of an inner code, as an approach tailored to large-scale support recovery. Still, their implementation and analysis are drastically different from results presented below, and the intended application scenarios lead to distinct modeling assumptions. Nevertheless, we deem it appropriate to point to these references and encourage the reader broadly interested in divide-and-conquer approaches to explore this related line of work.

Exact support recovery in noisy compressed sensing has been studied extensively in the literature [17]–[19]. In [17], the authors establish a connection between problem dimension, sparsity index, and the number of observations needed for exact support recovery using the LASSO algorithm. Furthermore, [18] and [19] offer necessary and sufficient conditions for exact support recovery in the presence of noise. A key result common among these studies is that $\mathcal{O}\left(k \log\left(p/k\right)\right)$ measurements are sufficient to recover a $k$-sparse vector of dimension $p$ in the presence of noise. Conventional compressed sensing solvers like LASSO [20] and iterative hard

thresholding [21] are known to achieve this scaling. However, most of these algorithms admit a computational complexity that scales as $\mathrm{poly}(p)$, which precludes the direct application of these schemes to the unsourced MAC problem.

Several works in the field of data stream computations, e.g., [22]–[24], aim to recover a $k$ out of $p$ sparse signal from a low-dimensional sketch. The algorithms therein feature measurement costs of $\mathcal{O}\left(k\log\left(p/k\right)\right)$ and computational costs of $\mathcal{O}\left(k\,\mathrm{polylog}(p)\right)$. However, these algorithms admit a failure probability that is bounded away from zero and, consequently, are not asymptotically reliable.

### B. Organization and Notation

The remainder of this article is organized as follows. In Section II, we describe the system model and introduce the various parameters used in this context. Section III provides detailed descriptions of the encoding and decoding operations employed within our proposed scheme. The performance of this system is analyzed in terms of error probability and average complexity in Section IV. In the process, an optimization framework is established to exploit the tension between these two considerations. Bounds on the average computational cost and the number of channel uses needed for our scheme to work reliably in the asymptotic regime where $K_\mathrm{a}, B \to \infty$ are derived in Section V. Simulation results are reported in Section VI to illustrate the performance of our scheme in the finite-block length regime. Finally, we conclude the paper in Section VII. Intricate proofs are relegated to the appendix.

At this stage, it is worth reviewing the notation we adopt throughout. We employ $\mathbb{R}_+$, $\mathbb{Z}_+$, and $\mathbb{N}$ to represent the non-negative real numbers, non-negative integers, and the natural numbers. For $a, b \in \mathbb{Z}_+$ with $a \leq b$, we use $[a:b]$ to denote $\{c \in \mathbb{Z}_+ : a \leq c \leq b\}$. For any $a, b \in \mathbb{R}_+$ with $a \leq b$, we use $[a, b]$ to denote $\{c \in \mathbb{R}_+ : a \leq c \leq b\}$. We employ $|A|$ for the cardinality of set $A$, and we use $[x]$ to designate the closest integer to $x$. We write $f(n) = \mathcal{O}(g(n))$ when there are constants $c$, $n_0$ such that $f(n) \leq cg(n)$ for all $n \geq n_0$.

## II. SYSTEM MODEL

Let $\mathbf{S}_\mathrm{tot}$ be the collection of devices within a network, and let $\mathbf{S}_\mathrm{a}$ denote the subset of active devices within a communication round, $\mathbf{S}_\mathrm{a} \subset \mathbf{S}_\mathrm{tot}$. We label the size of these sets by $|\mathbf{S}_\mathrm{tot}| = K_\mathrm{tot}$ and $|\mathbf{S}_\mathrm{a}| = K_\mathrm{a}$. Every active device wishes to communicate $B$ bits of information to a base station and, collectively, these data transfers must take place through an uncoordinated uplink transmission scheme. That is, transmissions are not scheduled centrally and, consequently, active devices must act independently of one another. The number of channel uses dedicated to this process is $N$, and we employ $W = \{\underline{w}_i : i \in \mathbf{S}_\mathrm{a}\}$ to represent the collection of $B$-bit message vectors associated with these active devices. In our proposed scheme, we assume that active devices pick their information message independently and uniformly at random from the set of binary sequences $\{0, 1\}^B$.

The base station facilitates a slotted structure for multiple access on the uplink through coarse synchronization. As such,

the signal available at the receiver assumes the form

$$\underline{y} = \sum_{i \in \mathbf{S}_\mathrm{a}} \underline{x}_i + \underline{z}, \tag{1}$$

where $\underline{x}_i$ is the $N$-dimensional vector transmitted by device $i$, and $\underline{z}$ represents additive white Gaussian noise with covariance $\sigma^2 \mathbf{I}$. As is often the case in similar settings, the signal sent by a device is power constrained, i.e., $\|\underline{x}_i\|_2^2 \leq NE_\mathrm{s}$ for $i \in \mathbf{S}_\mathrm{a}$, a scenario akin to [1]. The energy-per-bit of this system is defined as $\frac{E_b}{N_0} = \frac{NE_\mathrm{s}}{2B\sigma^2}$. Based on the observed signal, the receiver produces an estimate $\widehat{W}(\underline{y})$ for the list of transmitted binary vectors $W$ with $|\widehat{W}(\underline{y})| \leq K_\mathrm{a}$. Following the original formulation of [1], the per-user error probability defined by

$$P_\mathrm{e} = \frac{1}{K_\mathrm{a}} \sum_{i \in \mathbf{S}_\mathrm{a}} \Pr\left(\underline{w}_i \notin \widehat{W}(\underline{y})\right) \tag{2}$$

serves as a performance objective. In words, there is a penalty when a sent message is missed by the base station. Moreover, the total number of vectors in $\widehat{W}(\underline{y})$ is subject to a hard constraint, which prevents the base station from admitting an excessive number of guesses. The following key features distinguish this model from the traditional multiple access channel.

1) The population of users $K_\mathrm{tot}$ is very large and, hence, decoding becomes prohibitively complex when different users employ distinct codebooks. Consequently, we consider the scenario where all the active users employ a same codebook.
2) The decoder is tasked with providing an unordered list of messages.
3) Error probability is defined on a per user basis, as opposed to a success being contingent on identifying all the sent messages correctly.

We note that (1) can be expressed in matrix form as

$$\underline{y} = \mathbf{X}\underline{b} + \underline{z}, \tag{3}$$

where $\mathbf{X} \in \mathbb{R}^{N \times 2^B}$ denotes the common codebook and $\underline{b} \in \{0, 1\}^{2^B}$ is a binary vector that contains the indices of all the transmitted codewords. Under this viewpoint, the $i$th column $\mathbf{X}[:, i]$ represents the signal associated with message index $i$, and $\|\underline{b}\|_0 = K_\mathrm{a}$. The above formulation highlights a close connection between this viewpoint and a compressed sensing problem, with $\mathbf{X}$ playing the role of a sensing matrix and $\underline{b}$ being an unknown $K_\mathrm{a}$-sparse vector. As pointed out in the previous section, matrix $\mathbf{X}$ has $2^B$ columns with $B \approx 100$ and, hence, a naive CS algorithm cannot be employed to recover sparse vector $\underline{b}$. The dimensionality of this problem prevents the application of conventional algorithms without further modifications. Our goal is to devise an encoding and decoding scheme that achieves $P_\mathrm{e} \leq \varepsilon$, where $\varepsilon$ is a target error probability, and does so with manageable computational complexity. Table I summarizes the important parameters encountered in this article.

## III. PROPOSED SCHEME

The main idea to manage complexity consists in dividing the data stream generated by active devices into

TABLE I

THIS LIST CONTAINS KEY PARAMETERS ENCOUNTERED IN THE TREAT-
MENT OF CODED COMPRESSED SENSING

| Notation | Parameter Description |
|---|---|
| $K_{\text{tot}}$ | Total number of users in the system |
| $K_{\text{a}}$ | Number of active users |
| $B$ | Message length in bits |
| $N$ | Number of channel uses per round |
| $\varepsilon$ | Maximum tolerable probability of error per user |
| $n$ | Number of coded sub-blocks per round |
| $J$ | Number of coded bits per sub-block, |
| $M$ | Total number of coded bits, $M = nJ$ |
| $P$ | Total number of parity-check bits, $P = M - B$ |
| $\varepsilon_{\text{tree}}$ | Maximum probability of error for tree decoding |
| $C_{\text{cs}}$ | Computational complexity of CS sub-problem |
| $C_{\text{tree}}$ | Computational complexity of tree decoding |
| $b_j$ | Number of information bits in $j$th sub-block |
| $l_j$ | Number of parity bits in $j$th sub-block |
| $K$ | Size of output list for CS sub-problem |
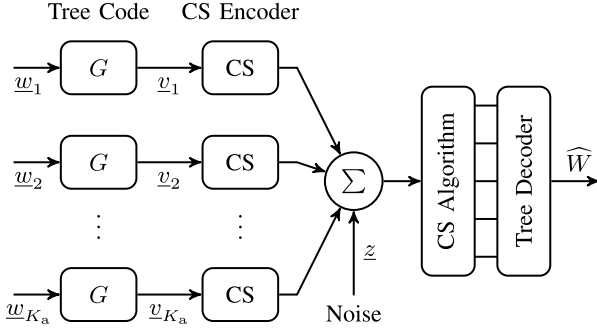


Fig. 1. This schematic diagram captures the overall architecture of the pro-
posed scheme. The information bits are split into sub-blocks, and redundancy
is added to individual components. Transmitted signals are then determined
via a CS matrix, and sent over the MAC channel. A CS algorithm recovers
the lists of sub-blocks, and a tree decoder reconstructs the original messages.

several sub-blocks. These sub-blocks, with their very short
packet fragments, are amenable to computationally efficient
compressed sensing (CS) algorithms. The transmission of
sub-blocks takes place sequentially, with every collection of
fragments becoming an instance of unsourced multiple access,
albeit one with a much reduced number of information bits.
Upon completion of these consecutive CS instances, the orig-
inal messages are recovered by stitching together compatible
fragments. Technically, this latter step is accomplished by
preemptively adding redundancy to the codewords, and then
leveraging this redundancy while combining sub-blocks via
a low-complexity tree-based algorithm. The specifics of this
envisioned scheme are detailed below. A notional diagram for
the proposed system appears in Fig. 1.

### A. Encoding Process

The transmission strategy features two complementary com-
ponents: a systematic linear block code based on random parity
checks, which we call tree code, and a signaling scheme rooted
in compressed sensing. These building blocks are detailed
below, and a top level overview of the encoding process is
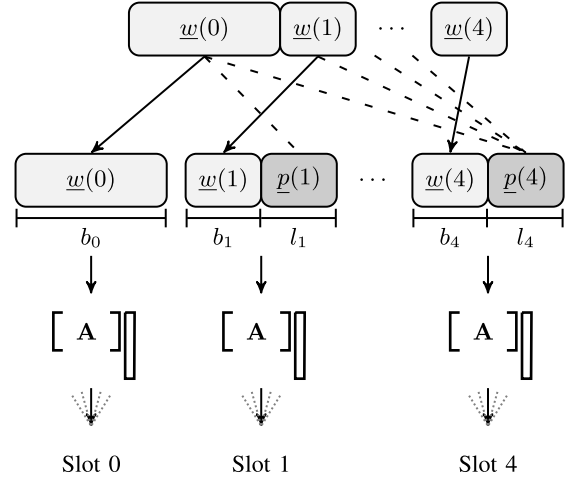illustrated in Fig. 2.



Fig. 2. Encoding for CCS proceeds as follows. Information bits are
partitioned into $n$ fragments. These fragments are enhanced with redundancy
in the form of parity bits. Each sub-block is converted into a signal via a CS
matrix, and subsequently transmitted over a time slot.

*1) Tree Encoding:* Every $B$-bit binary message $\underline{w}$ is parti-
tioned into $n$ sub-blocks, where the $j$th sub-block consisting of
$b_j$ message bits, with $\sum_{j=0}^{n-1} b_j = B$. Under this fragmentation,
we can express a message as $\underline{w} = \underline{w}(0)\underline{w}(1)\cdots\underline{w}(n-1)$.
The tree encoder appends $l_j$ parity check bits to sub-block $j$,
bringing the total length of every sub-block to $b_j + l_j = J =
M/n$ bits. The first block is chosen to have $b_0 = J$ information
bits and, necessarily, $l_0 = 0$ parity bits. For subsequent sub-
blocks, the parity bits are constructed as follows. Let $\underline{p}(j)$
denote the parity bits in sub-block $j$. These $l_j$ checks are
selected to satisfy random parity constraints associated with
the message bits that are contained in preceding sub-blocks.
Mathematically,

$$\underline{p}(j) = \sum_{\ell=0}^{j-1} \underline{w}(\ell) G_{\ell,j-1}$$

where $G_{\ell,j-1}$ is a $m_\ell \times l_j$ random binary matrix. That is,
the entries in $G_{\ell,j-1}$ are independent Bernoulli trials, each
with parameter half. Parity bits are computed using modulo-
2 arithmetic and, as such, they remain binary. These parity bits
are subsequently injected into a codeword $\underline{v}$, which assumes
the form

$$\underline{v} = \underbrace{\underline{w}(0)}_{\underline{v}(0)} \underbrace{\underline{w}(1)\underline{p}(1)}_{\underline{v}(1)} \cdots \underbrace{\underline{w}(n-1)\underline{p}(n-1)}_{\underline{v}(n-1)}. \qquad (4)$$

In effect, $\underline{p}(j)$ acts as parity constraints for random collections
of information bits preceding fragment $j$. In Section IV-C,
we describe an optimization framework for the selection of
parity lengths $l_1, l_2, \ldots, l_{n-1}$.

*2) CS Encoding:* Let $\mathbf{A} = [\underline{a}_1, \ldots, \underline{a}_{2^J}] \in \{\pm\sqrt{E_\text{s}}\}^{\frac{N}{n} \times 2^J}$
be a compressed sensing matrix that is designed to recover
$K_\text{a}$-sparse binary vectors from $N/n$ noisy observations with a
low probability of failure. This matrix is meant to operate on
tree-coded sub-blocks, which explains its size. Conceptually,
every possible tree-coded $J$-bit vector is mapped to a column
of $\mathbf{A}$. Thus, we can view the columns of $\mathbf{A}$ as the set of
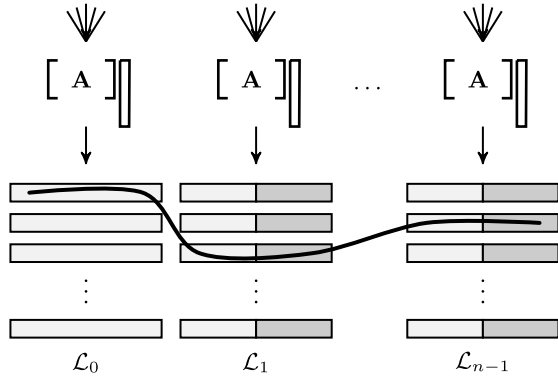
Fig. 3. The decoding process at the destination starts by running a CS recovery algorithm on the noisy signal aggregate corresponding to each time slot. This yields lists of coded fragments $\mathcal{L}_0, \mathcal{L}_1, \ldots, \mathcal{L}_{n-1}$, one list per time slot. Message fragments are then stitched together using the redundant structure of the tree code, as depicted in this illustration.

potentially transmitted signals by a user over the duration of a CS sub-block. In Fig. 2, the CS encoding process is portrayed as the product of an abstract sensing matrix and a tall index vector of length $2^J$. Under this representation, the binary index vector $\underline{r}_i(j)$ associated with $\underline{v}_i(j)$, the codeword fragment of user $i$, has zeros everywhere except at location $\mathrm{decimal}(\underline{v}_i(j))$, where $\mathrm{decimal}(\underline{x})$ denotes the radix ten equivalent of the binary vector $\underline{x}$. The figure also emphasizes the divide-and-conquer approach where the processing of one message leads to the creation of $n$ CS instances. During every transmission round, the multiple access channel sums up the contributions of all the active users. This is equivalent to sending aggregate signal $\mathbf{A}\underline{r}(j)$ over a noisy channel, where $K_a$-sparse vector $\underline{r}(j) = \sum_{i \in \mathbf{S}_a} \underline{r}_i(j)$.

### B. Decoding Process

As mentioned above, the input to the decoding process is the sum of the signals transmitted by the active users plus noise. This received signal is segmented along the boundaries of the fragments, with the portion corresponding to each slot viewed as a sub-block. Not too surprisingly, the decoding strategy also consists of two components: a CS recovery algorithm that operates over every sub-block, and a tree decoder operating across sub-blocks. These steps are illustrated in Fig. 3, and they are detailed below.

*1) CS Decoding:* The aggregate signal received at the base station during the $j$th sub-block can be expressed as $\underline{y}(j) = \mathbf{A}\underline{r}(j) + \underline{z}(j)$, where $\underline{r}(j) \in \{0,1\}^{2^J}$ is a $K_a$-sparse binary vector that incorporates the indices of the $j$th sub-blocks transmitted by the active users. The task of the CS decoder is to provide an estimate of the sparse vector $\underline{r}(j)$ based on the received signal $\underline{y}(j)$ during the corresponding time slot. This is accomplished by first applying a conventional CS recovery algorithm, e.g., non-negative least squares (NNLS) or LASSO, to get an estimate $\hat{\underline{r}}^{(\mathrm{cs})}(j)$ of vector $\underline{r}(j)$. Yet, this does not ensure that the entries of vector $\hat{\underline{r}}^{(\mathrm{cs})}(j)$ are binary. The desired binary estimate $\hat{\underline{r}}(j)$ is obtained by setting the $K$ largest entries of vector $\hat{\underline{r}}^{(\mathrm{cs})}(j)$ to one and the remaining $2^J - K$ entries to zero. Sparsity parameter $K$ is chosen as $K = K_a + K_\delta$, where $K_\delta$ is a small positive integer. The fragments corresponding

to index vector $\hat{\underline{r}}(j)$ are aggregated on a list $\mathcal{L}_j$, which acts as the output of the CS algorithm for this sub-problem. Although the list output by the CS recovery algorithm is larger than $K_a$, the quantity $K_\delta$ is carefully chosen such that erroneous fragments remain very unlikely to satisfy the parity check constraints associated with the tree code.

*2) Tree Decoding:* The tree decoder seeks to recover the original messages transmitted by all the users by piecing together valid sequences of elements drawn from the various CS lists. To this end, the access point constructs a decoding tree for *every candidate message* as follows. We fix a sub-block from $\mathcal{L}_0$, the list of all possible initial sub-blocks supplied by the CS decoder. We view this fragment as the root node of a tree. Once this sub-block is selected, there are $K$ possible choices for the subsequent sub-block, and these are the nodes which appear on $\mathcal{L}_1$. Similarly, there are $K$ possible choices for the second sub-block for each partial path identified thus far and, hence, there are $K^2$ partial paths at stage two. This process continues until stage $(n-1)$ is reached. At this point, the tree features $K^{n-1}$ leafs. Every path connecting the root node to a leaf becomes a candidate message, yet invalid paths that do not meet their own parity requirements can be eliminated. After this consistency check, if there remains a single valid path that meets its own parity checks, then the decoder outputs the corresponding message; otherwise, it reports a failure.

We emphasize that the number of possible paths increases exponentially with the stages of the tree and, hence, a naive search through all the leaf nodes is unrealistic. In practice, invalid paths can be pruned iteratively through the parity check constraints. Specifically, at stage $j \geq 1$, the decoder retains only nodes that satisfy the $l_j$ parity constraints on all the message bits preceding that stage. This iterative procedure continues until the last stage is reached. The complexity of this decoding scheme depends on the number of nodes surviving each stage because parity checks only need to be enforced on the children of surviving nodes in subsequent stages of the tree decoding process. Figure 4 gives a step-by-step description of the various stages involved in the tree decoding algorithm.

*3) Iterative Extension:* The CCS framework admits an iterative extension based on successive cancellation. At the end of the tree decoding process, the receiver has identified a collection of candidate messages $\widehat{W}(\underline{y})$. The indices of the $j$th fragments of these messages necessarily appear on $\hat{\underline{b}}(j) \in \{0,1\}^{2^B}$. However, when a sub-block is mistakenly discarded by the CS recovery algorithm or if the tree decoder fails to stitch a message without including extraneous paths, the number of used fragments within $\hat{\underline{b}}(j)$ may not be equal to $K_a$. In particular, on a first pass, $|\widehat{W}(\underline{y})|$ may be less than $\|\underline{b}(j)\|_0 = K_a$. This situation invites an iterative procedure to recover the missing fragments and messages, when the list size output by the tree decoder $|\widehat{W}(\underline{y})|$ is less than $K_a$. The contributions of the fragments associated with $\widehat{W}(\underline{y})$ can be subtracted from their corresponding CS sub-signal $\underline{y}(j)$ to yield
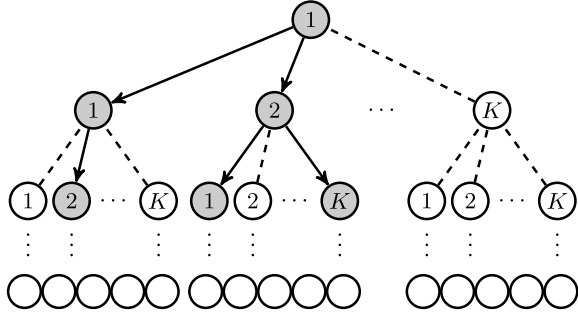
$$\underbrace{\underline{y}(j) - \mathbf{A}\hat{\underline{b}}_{\mathrm{tree}}(j)}_{\underline{y}_{\mathrm{residual}}(j)} = \underbrace{\underline{x}(j) - \mathbf{A}\hat{\underline{b}}_{\mathrm{tree}}(j)}_{\underline{x}_{\mathrm{residual}}(j)} + \underline{z}(j) \quad (5)$$
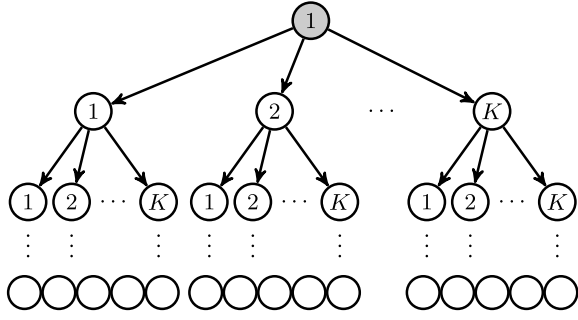
(a) Stage 0: Processing one element from $\mathcal{L}_0$ at a time, a fragment is selected as the root node of a tree.
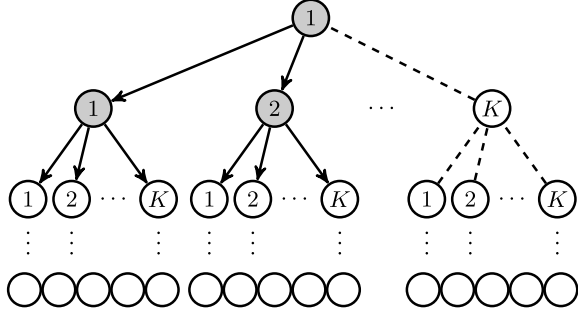


(b) Stage 1: Fragments in $\mathcal{L}_1$ act as the children of the root node. Parity requirements are checked and only complying nodes, nodes 1 and 2 highlighted in the figure, are retained.



(c) Subsequent Stages: Candidate fragments from a subsequent stage become the children of complying nodes. Again, parity constraints are verified for every child and the tree is pruned accordingly, as highlighted above.
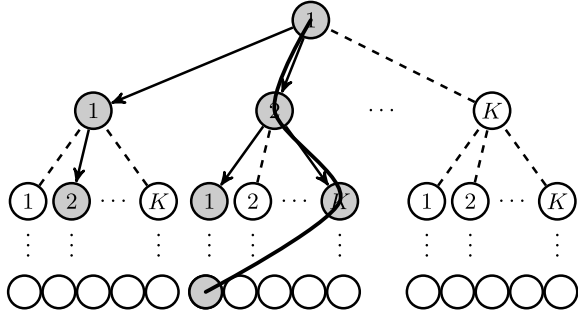


(d) Last Stage: Parity constraints are verified at the leafs. A valid message on the CS tree will survive, but decoding is successful only if no other paths meet its parity requirements. We highlight the legitimate path in black above.

Fig. 4. These images illustrate the operation of the tree decoding algorithm across sub-blocks.

where $\hat{\underline{b}}_{\text{tree}}$ only contains the indices within $\hat{\underline{b}}(j)$ that are associated with fragments found on the output of the tree decoder $\widehat{W}(\underline{y})$. The above equation resembles the standard

form of the noisy compressed sensing problem, where $\mathbf{A}$ is the sensing matrix and $\underline{b}(j) - \hat{\underline{b}}_{\text{tree}}(j)$ is the unknown binary vector with sparsity index $K_{\text{a}} - |\widehat{W}(\underline{y})|$. An improved estimate for vector $\underline{b}(j)$ can then be obtained by solving the above residual problem using the divide-and-conquer approach where the CS recovery algorithm operates at sub-block level and the tree decoder across sub-blocks. The list size output by the CS algorithm at the sub-block level for this iteration is reduced accordingly, based on the number of unidentified messages $K_{\text{a}} - |\widehat{W}(\underline{y})|$. This successive interference cancellation method can be repeated iteratively, potentially leading to significant gains in performance, especially for the first few iterations.

## IV. PERFORMANCE ANALYSIS

In this section, we study the performance of the CCS multiple access scheme. We rely mostly on established results to characterize the CS component of the scheme. On the other hand, the analysis of tree coding is new and, as such, it is our primary focus. In the remainder of this section, we assume that the list size output by the CS decoder is set to $K = K_{\text{a}}$. Though not trivial, it is possible to extend the analysis by relaxing this assumption to accommodate the more general case $K > K_{\text{a}}$.

Let $E_i$ denote the event that the message sent by user $i$ is not resolved by the tree decoder. This occurs when at least one sub-block generated by this user is not present on the lists output by the CS decoder, or when the tree decoder outputs more than one valid message for this user. Let $C_i$ be the event that all the sub-blocks generated by this user are present on the lists output by the CS decoder. With this notation, we can write probability $\Pr(E_i)$ as

$$\Pr(E_i) = \Pr(E_i|C_i)\Pr(C_i) + \Pr(E_i|\overline{C_i})\Pr(\overline{C_i}). \quad (6)$$

Consider the event where sub-block $i_j$ transmitted by user $i$ during slot $j$ fails to appear on the list output by the CS algorithm at that stage. We note that $p_{\text{cs}}$, the probability of this event, is uniformly the same among users and across time slots due to the symmetric structure of the decoding process. Equivalently, the probability that the list output by the CS recovery algorithm contains sub-block $i_j$ is equal to $1 - p_{\text{cs}}$. When the CS recovery algorithm fails to identify at least one of the sub-blocks belonging to a particular user, then the output of the tree decoder may not contain the original codeword sent by that user. Given that the CS sub-blocks are decoded independently, $\Pr(C_i)$ can be computed as $\Pr(C_i) = (1 - p_{\text{cs}})^n$.

We write the conditional event that the tree decoder declares a failure because of more than one path surviving the tree decoding process as $E_i|C_i$, and we represent the probability corresponding to this event by $p_{\text{tree}}$. In the base form of the algorithm, without the optional iteration process discussed in Section III-B3, the quantity $P_e$ is the same as $\Pr(E_i)$. They can be computed using (6), along with the aforementioned observations,

$$P_e = p_{\text{tree}}(1 - p_{\text{cs}})^n + (1 - (1 - p_{\text{cs}})^n)$$
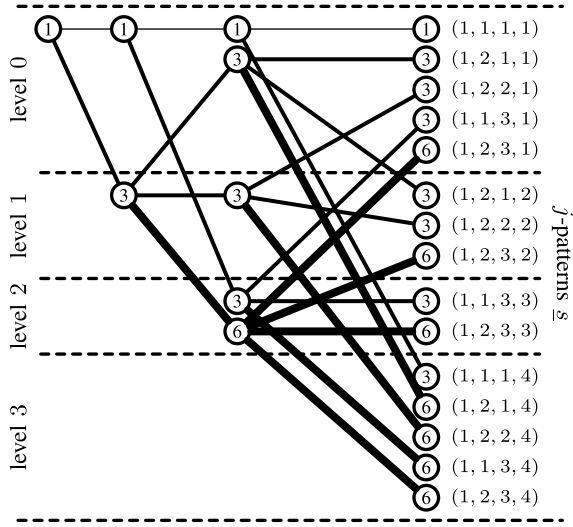$$= 1 - (1 - p_{\text{tree}})(1 - p_{\text{cs}})^n. \quad (7)$$

Fig. 5. This diagram illustrates the level structure of tree coding and the reduction afforded by $j$-patterns. In this particular scenario, the total number of paths to consider decreases from 64 to 15. The number of (partial) paths $n(\underline{s})$ in each equivalence class $\underline{s}$ appears inside the corresponding node, and it is also highlighted through the thickness of the connection.

In view of this expression, we turn our attention to better understanding $p_{\text{tree}}$.

## A. Tree Code Analysis

The exact analysis of the tree decoding process, although possible, is quite intricate. Herein, we present the key results, but relegate details to Appendix A. The main idea is to compute the expected number of surviving erroneous paths at various levels of the tree decoding process, and then bound the probability that erroneous paths survive altogether using the Markov inequality. Let $L_j$ denote the random variable corresponding to the number of erroneous paths that survive stage $j \in [1 : n-1]$ of the tree decoding process, assuming that all the fragments corresponding to the root sub-blocks appear on the CS output lists and that the list size is $K = K_a$. The expected value of this random variable is characterized in Theorem 1.

*Theorem 1:* Starting from a fixed root node, the expected number of erroneous surviving paths at level $j-1$ is given by

$$\mathbb{E}[L_{j-1}] = \sum_{\underline{s} \in \mathcal{P}_j} n(\underline{s}) \Phi_{\underline{s}}\left(\frac{1}{2}\right) - 1, \quad j \in [1:n].$$

where $\mathcal{P}_j$ represents the set of $j$-patterns (OEIS A008277),

$$n(\underline{s}) = (K-1)(K-2)\cdots(K-(d(\underline{s})-1))$$

with $d(\underline{s})$ denoting the number of distinct integers in $\underline{s}$. Furthermore, $\Phi_{\underline{s}}(x)$ is the probability generating function of $T(\underline{s})$, the number of statistically discriminating parity bits for a path associated with pattern $\underline{s}$. It takes the form

$$\Phi_{\underline{s}}(x) = \mathbb{E}\left[x^{T(\underline{s})}\right] = \sum_{t=0}^{l_1+\cdots+l_{j-1}} \Pr(T(\underline{s}) = t)x^t$$

where $\Pr(T(\underline{s}) = t)$ is the probability that exactly $t$ parity bits are statistically discriminating for $j$-pattern $\underline{s}$.

*Proof:* A proof for this theorem is available in Appendix A. Section A-B presents conditions for erroneous paths to survive. A complexity reduction scheme based on $j$-patterns and the Bell numbers is found in Section A-C. The notion of statistically discriminating parity bits appears in Section A-D, along with a characterization of their probability distributions. Collectively, these components provide a computationally tractable means to compute $\mathbb{E}[L_{j-1}]$ for the parameters of interest. ∎

Figure 5 illustrates the reduction in complexity afforded by $j$-patterns for $n = 4$ and $K = 4$. A candidate path enters a new level whenever its next fragment comes from a message that has not been visited before. As a corollary to this Theorem 1, we get a bound on $p_{\text{tree}}$. This bound, presented in Corollary 2, is particularly close when $p_{\text{tree}}$ is small.

*Corollary 2:* Assuming that all the fragments of a message appear on their respective CS lists, the probability that the tree decoder declares a failure due to more than one path surviving is bounded by

$$p_{\text{tree}} \leq \mathbb{E}[L_{n-1}].$$

*Proof:* We can bound the probability of at least one extraneous path surviving by

$$p_{\text{tree}} = \Pr(E_i|C_i) = \Pr(L_{n-1} \geq 1)$$
$$\leq \mathbb{E}[L_{n-1}].$$

The inequality follows from an application of the Markov inequality. ∎

This bound can be substituted in (7) to provide guarantees on the performance of tree coding. We illustrate its use in our upcoming numerical results. Still, the amount of computations involved in applying Theorem 1 remains significant. Above, the derivations are presented for a given parity allocation $l_1, l_2, \ldots, l_{n-1}$. Moving forward, we need to find suitable values for these parameters. In principle, this latter task involves repetitive computations of $p_{\text{tree}}$ for numerous possible assignments. To circumvent the computational load this would entail, we develop an easily computable approximation for $p_{\text{tree}}$, which enables the selection of appropriate values for $l_1, \ldots, l_{n-1}$. This is accomplished below.

## B. Approximate Tree Code Analysis

The complexity associated with the exact characterization of tree codes stems from the fact that many cases must be considered separately. These cases correspond to equivalence classes associated with $j$-patterns, as discussed above. From an abstract point of view, three confounding factors make the performance analysis of tree coding challenging. Several fragments within an erroneous candidate codeword may come from a same message. When $\underline{w}_i(j)$ and $\underline{w}_i(q)$ are both part of an erroneous candidate, with $j < q$, the probability that the parity bits of the later fragment remain statistically discriminating is reduced. Moreover, although messages must be distinct overall, two different messages can share a same fragment. Mathematically, $\underline{w}_i \neq \underline{w}_k$ does not imply $\underline{w}_i(j) \neq \underline{w}_k(j)$ for all $j \in [0 : n-1]$. When fragments across messages are identical, they produce a loss in discrimination power in

the corresponding parity bits. This also introduces correlation in error events across time slots.

As mentioned above, an exact analysis may be suitable for performance characterization, but an approximate analysis is desirable for parameter selection. To facilitate the latter, we make a simplifying assumption about the nature of erroneous paths. A partial candidate vector

$$\underline{w}_{i_0}(0)\underline{w}_{i_1}(1)\cdots\underline{w}_{i_j}(j)$$

is termed erroneous if not all indices $i_0, i_1, \ldots, i_j$ come from a same message. That is, there exists at least one $q < j$ such that $i_q \neq i_j$. Identifying erroneous vectors is the main purpose of the tree decoder. We state our simplifying assumption that renders analysis more tractable formally below for future reference.

*Assumption 3 (Simplifying Structure):* The collection of messages sent to the destination is such that, for any $i \neq k$,

$$\underline{w}_i(j) \neq \underline{w}_k(j) \tag{8}$$

for all $j \in [0 : n-1]$ where $b_j > 0$.

We point out that the condition in (8) is met with high probability at every stage $j$ where $b_j$ is large. Also, we emphasize that it has a greater impact in earlier stages where $b_j$ tends to be large. Given (8), the events that erroneous partial candidate vectors fulfill their parity constraints are pairwise independent. A stronger notion of independence is obtained when the collection of erroneous partial candidate vectors form a linearly independent set. For parameters of interest, this is nearly achieved in that the dimensionality of the span of the erroneous partial candidate vectors is typically close to the number of such vectors. In any case, we conduct a mean analysis below, which circumvents technical issues related to independence. As we will see shortly, the level of performance predicted under this approximation appears accurate for parameters of interest. However, we also note that one can create idiosyncratic cases where the predicted performance under Assumption 3 is overly optimistic. We employ $\tilde{L}_j$ to denote the approximate number of erroneous paths that survive stage $j$, computed under Assumption 3. This makes the approximate results easy to distinguish from the true $L_j$ introduced in Section IV-A.

*Proposition 4:* Under Assumption 3, the expected values of $\tilde{L}_j$ can be computed as

$$\mathbb{E}\big[\tilde{L}_j\big] = \sum_{q=1}^{j}\left(K^{j-q}(K-1)\prod_{\ell=q}^{j}p_\ell\right) \tag{9}$$

where $p_\ell = 2^{-l_\ell}$ and $j \in [1 : n-1]$.

*Proof:* A proof is available in Appendix B. ∎

Having gained a handle on the expected growth in the number of erroneous paths, we are in a position to analyze the complexity of the tree decoder. We define the computational complexity $C_{\text{tree}}$ of the tree decoder as the total number of parity check constraints that need to be verified. The expected value of this quantity is captured in Proposition 5.

*Proposition 5:* Under Assumption 3, a closed-form expression for the expected computational complexity $\mathbb{E}[C_{\text{tree}}]$ is

$$\mathbb{E}[C_{\text{tree}}] = K\left(\sum_{j=1}^{n-1}l_j + \sum_{j=1}^{n-2}l_{j+1}\sum_{q=1}^{j}K^{j-q}(K-1)\prod_{\ell=q}^{j}p_\ell\right) \tag{10}$$

where $p_\ell = 2^{-l_\ell}$.

*Proof:* Each non-leaf node that survives stage $j$ engenders $K$ children. For every such child, a total of $l_{j+1}$ parity checks must be verified. Hence, the computational complexity and its expected value can be expressed as

$$C_{\text{tree}} = Kl_1 + \sum_{j=1}^{n-2}(L_j+1)Kl_{j+1} \tag{11}$$

$$\mathbb{E}[C_{\text{tree}}] = Kl_1 + \sum_{j=1}^{n-2}\left(\mathbb{E}[L_j]+1\right)Kl_{j+1}. \tag{12}$$

Using the approximate expression for $L_j$ found in (9) and substituting it into (12), we get the closed-form expression of (10). ∎

A related quantity, which we call approximate tree complexity, is the number of tree nodes on which parity check constraints must be verified. This latter quantity is less accurate, as it does not take into consideration that distinct nodes may have a different number of parity bits. However, this looser definition facilitates mathematical tractability in certain cases and, as such, we characterize its expected value below.

*Corollary 6:* Under Assumption 3, the expected number of nodes for which parity checks must be computed is given by

$$\mathbb{E}\big[\tilde{C}_{\text{tree}}\big] = (n-1)K + K\sum_{j=1}^{n-2}\sum_{q=1}^{j}K^{j-q}(K-1)\prod_{\ell=q}^{j}p_\ell \tag{13}$$

where $p_\ell = 2^{-l_\ell}$.

*Proof:* As before, every non-leaf node that survives stage $j$ will produce $K$ children. Thus, the number of nodes on which parity checks must be performed, and its expected value are

$$\tilde{C}_{\text{tree}} = (n-1)K + \sum_{j=1}^{n-2}L_j K \tag{14}$$

$$\mathbb{E}\big[\tilde{C}_{\text{tree}}\big] = (n-1)K + \sum_{j=1}^{n-2}\mathbb{E}[L_j]K. \tag{15}$$

Substituting the approximate expression for $L_j$ found in (9) into (15), we get (13). ∎

Figure 6 offers supportive evidence to the fact that the expected number of surviving paths predicted under Assumption 3 is reasonably accurate for parameters of interest. This plot shows the exact values computed using Theorem 1, the optimistic approximation of Proposition 4, and the results obtained via numerical simulations. Reported findings are obtained under the following system parameters: $K = 200$, $n = 11$, $J = 15$. The parity allocation is $(l_1, l_2, l_3, \ldots, l_{10}) = (6, 8, 8, \ldots, 8, 13, 15)$. Similar findings can be obtained for a range of system parameters.
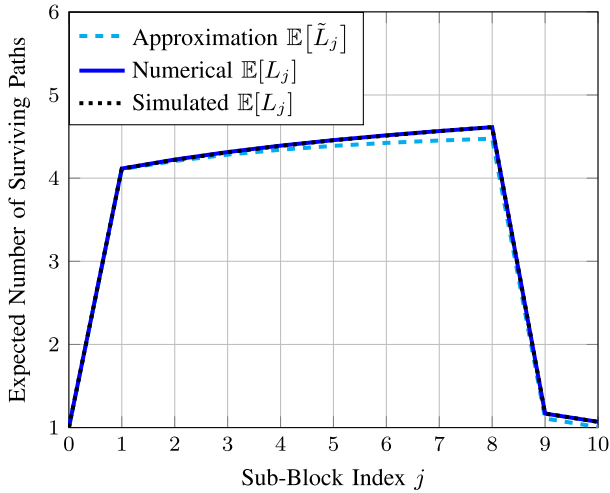
Fig. 6. This plot shows the expected number of surviving paths at various stages during the tree decoding process. The approximate values derived under simplifying Assumptions 3 remain close to the exact values over a range of system parameters.

## C. Allocating Parity Checks

The rate of the tree code is intrinsically determined by the number of parity constraints added to the information bits. Once the rate is fixed, one must decide where to position these parity bits. Allocating more parity bits towards the initial stages of the tree will limit expected complexity, as these parity checks will prune the tree early. However, this comes at the expense of a higher probability of error. On the other hand, pushing parity bits towards later stages will have the dual effect of reducing the probability of error and increasing computational complexity. Hence, the location of parity checks should be selected judiciously as a means to tradeoff performance and complexity.

To this end, we formulate the constrained optimization problem of minimizing the expected complexity subject to the probability of failure being less than a carefully chosen threshold $\varepsilon_{\text{tree}}$:

$$\underset{(l_1,\ldots,l_{n-1})}{\text{minimize}} \quad \mathbb{E}[C_{\text{tree}}] \tag{16}$$

$$\text{subject to} \quad p_{\text{tree}} \leq \varepsilon_{\text{tree}} \tag{17}$$

$$\sum_{j=1}^{n-1} l_j = M - B \tag{18}$$

$$l_j \in [0:J] \quad \forall j \in [1:n-1]. \tag{19}$$

The last condition ensures that $l_j$ is a non-negative integer that does not exceed the length of a sub-block.

This optimization problem is non-linear and it is, in general, difficult to solve. As such, we alter the objective function and relax constraints. Drawing inspiration from the Markov inequality, we replace $p_{\text{tree}} \leq \varepsilon_{\text{tree}}$ by $\mathbb{E}[L_{n-1}] \leq \varepsilon_{\text{tree}}$. We also relax the integer requirements to $(l_1,\ldots,l_{n-1}) \in \mathbb{R}_+^{n-1}$. A first attempt at finding a tractable objective function consists in adopting the expected complexity $\mathbb{E}[C_{\text{tree}}]$ found in (10), which is derived under Assumption 3. Still, the ensuing problem remains challenging because of the non-convex nature

of the objective function. Hence, instead of minimizing the average number of parity check constraints that need to be verified, we minimize the average number of nodes on which parity check evaluations must be performed, as in Corollary 6. After these modifications, the optimization problem for allocating parity lengths can be posed as follows:

$$\underset{(p_1,\ldots,p_{n-1})}{\text{minimize}} \quad \mathbb{E}[\tilde{C}_{\text{tree}}] \tag{20}$$

$$\text{subject to} \quad \mathbb{E}[\tilde{L}_{n-1}] \leq \varepsilon_{\text{tree}} \tag{21}$$

$$\sum_{j=1}^{n-1} \log_2\left(\frac{1}{p_j}\right) = M - B \tag{22}$$

$$p_j \in \left[\frac{1}{2^J}, 1\right] \quad \forall j \in [1:n-1]. \tag{23}$$

Above, we express the optimization problem in terms of $p_1,\ldots,p_{n-1}$ to emphasize the fact that this problem is a geometric program [25]. Given this structure, an optimal solution can be attained using standard convex solvers. In implementing the proposed scheme, we adopt a parity allocation based on a quantized version of the unconstrained solution to this relaxed optimization. Specifically, if $p_1^*,\ldots,p_{n-1}^*$ is the optimal solution to the aforementioned problem, then we allocate parity check bits as $l_j = \left\lceil \log_2\left(1/p_j^*\right) \right\rceil$ for $j \in [1:n-1]$.

Empirically, the solutions obtained using this technique perform well. For a specific scenario, one could seek to improve performance by running a local search around this suggested operating point using the exact expressions derived in Section IV-A as a performance criterion. Still, in view of the adequate performance achieved using the relaxed optimization, this more involved approach seems unnecessary to capture most of the benefits of tree coding. Intuitively, adding more parity-check bits towards the end of the tree is beneficial from an error performance perspective. Even if the true transmitted path is the lone survivor of tree decoding until the penultimate stage, there are still $K_a$ possible continuations for the final stage. Thus the probability of error is at least $K_a 2^{-l_n}$ and this is smallest when $l_n$ takes its largest possible value, which is $J$. For any reasonable complexity and performance constraints, the results for the optimization problem developed in this section are consistent with this intuition as can be seen from simulation results.

## V. Asymptotic Analysis

As is often the case with problems of inference and communication systems, we seek additional insights by studying the CCS algorithm in the context of large settings. In particular, we explore the average computational complexity and the number of channel uses needed for CCS to be asymptotically reliable. Throughout this section, we assume that the number of active users $K_a$ and the message length $B$ jointly tend to infinity. We examine two relevant regimes for parameters $K_a$ and $B$, which will be referred henceforth as the logarithmic regime and the linear regime, respectively: (i)
  1) $B = \alpha \log_2 K_a$ for some fixed constant $\alpha > 1$,
  2) $B = \varrho K_a$ for some fixed constant $\varrho > 0$.
The scaling in the logarithmic regime applies, for instance, to scenarios akin to [8] where every active user sends its

identity as data. When the number of active users $K_a$ scales sub-linearly with the total number of users $K_{tot}$, i.e., $K_a = K_{tot}^{1/\alpha}$ for $\alpha > 1$, each active user needs to send $B = \log_2 K_{tot} = \alpha \log_2 K_a$ bits to uniquely identify themselves. On the other hand, the linear regime corresponds to situations where the payload is proportional to the number of active users. For ease of exposition, we focus on the special case $B = K_a$ in regime (ii). Nevertheless, the results we derived are valid for any linear factor $\varrho > 0$. We remark that these regimes are selected, partly, to enable the comparison of our algorithm with prior art.

From Section IV-C, we already know that the positioning of parity bits can act as a mechanism to tradeoff performance and computational complexity. As mentioned before, allocating parity bits towards the later stages of CCS improves performance at the expense of complexity; whereas placing check bits in early stages has the reverse effect. In view of this principle, we consider two distinct parity bit allocation strategies for these asymptotic regimes, one for each of the extreme behaviors outlined above. Specifically, we consider a scenario where all the parity bits are located towards the end of the codewords and we study the situation where the parity bits are equally distributed among all the non-root sub-blocks. In this context, we provide bounds for the number of parity bits, the number of channel uses, and the average computational complexity needed to achieve asymptotic reliable communication ($P_e \to 0$ as $K_a \to \infty$). This is accomplished for the trailing and uniform parity allocations in both regime (i) and regime (ii), leading to four distinct cases.

For the two parity profiles mentioned above, we show that the number of channel uses (i.e., the sample complexity in the compressed sensing literature) scales in an order-optimal fashion with respect to $K_a$ and $B$. The average computational complexity scales sub-linearly with the dimension of the original compressed sensing problem, $2^B$, in both cases. We also demonstrate that, although the number of channel uses is order optimal in both scenarios, the pre-constant in the Big-$\mathcal{O}$ terms is larger when parity bits are uniformly distributed across fragments than when they are placed at the end. This advantage in terms of channel uses in the latter case comes at the expense of a higher average computational complexity. These findings are stated formally in Theorem 8 and Theorem 9. Throughout the paper, we use order notation to express the dependence of the number of channel uses and decoding complexity of the compressed sensing part on the number of active users. We omit the pre-constant which depends, partly, on the signal to noise ratio.

*Remark 7:* It may be useful to reiterate that the application of conventional CS solvers (e.g., LASSO, OMP) to the global uncoordinated MAC problem of dimension $2^B$ entails a computational complexity that scales polynomially (at least linearly) with $2^B$. Specifically, the complexity for such an approach is on the order of $\mathcal{O}(2^{\gamma B})$ for some $\gamma \geq 1$. Moreover, the number of channel uses needed for this scheme to work is equivalent to the optimal number of samples needed to solve a $K_a$-sparse problem of dimension $2^B$, which is $\mathcal{O}(K_a \log_2 2^B)$. Thus, the optimal number of samples scales as $\mathcal{O}(K_a \log_2 K_a)$ in regime (i), and $\mathcal{O}(K_a^2)$ in regime (ii).

TABLE II

THIS OFFERS A SUMMARY OF THE PARAMETERS FOR THE TWO SCALING REGIMES

|   | Regime (i) | Regime (ii) |
|---|---|---|
| $B$ | $\alpha \log_2 K_a$ | $K_a$ |
| $J$ | $c_1 \log_2 K_a$ | $c_1 \log_2 K_a$ |
| $n$ | $c_2$ | $c_2 \frac{K_a}{\log_2 K_a}$ |
| $P$ | $(c_1 c_2 - \alpha) \log_2 K_a$ | $(c_1 c_2 - 1) K_a$ |

While the conventional approach is impractical for unsourced multiple access, these theoretical scaling behaviors serve as benchmarks for the asymptotic analysis of CCS.

Proceeding forward, the various parameters employed in our asymptotic analysis are chosen as follows. The list size output by the CS algorithm at the sub-block level is set to $K = K_a$. The length of each coded sub-block is selected to be $J = c_1 \log_2 K_a$ for fixed constant $c_1 > 1$ (with $c_1 < \alpha$ in regime (i)). We note that $c_1$ can be chosen arbitrarily close to one from the right. With this choice, the sparsity index admits a sub-linear scaling in the dimension of the compressed sensing problems at the sub-block level; that is, $K_a/2^J = K_a^{-(c_1-1)} \to 0$ as $K_a \to \infty$. We take the outer code to have a fixed rate, with $nJ = \Theta(B)$. In turn, this yields $n = \Theta(B/J)$ and, hence, the number of sub-blocks is of the form $n = c_2$ and $n = c_2 \frac{K_a}{\log_2 K_a}$ for regimes (i) and (ii), respectively. We stress that constant $c_2$ is implicitly determined by $c_1$ and the total number of parity bits. This information is summarized in Table II.

From (7), we get an upper bound on the overall error probability,

$$P_e \leq n p_{cs} + p_{tree}. \qquad (24)$$

Thence, asymptotic reliability is guaranteed whenever $n p_{cs}$ and $p_{tree}$ both decay to zero as $K_a \to \infty$. In the upcoming analysis, we show that the former condition is obtained by allocating the required number of channel uses needed to solve a $K_a$ out of $2^J$ sparse problem for the CS component of the scheme. Once this is accomplished, it suffices to identify conditions under which $p_{tree}$ goes to zero asymptotically to get the desired results. This proof strategy is applied to the two scenarios of interest below.

### A. Parity Bits Located at the End of Codewords

Let $C_{cs}$, $\mathbb{E}[C_{tree}]$, $\mathbb{E}[C]$, and $N$ denote the computational complexity of solving the $n$ CS sub-problems, the average tree decoding complexity, the average overall decoding complexity, and the total number of channel uses, respectively. Then, the following theorem holds.

*Theorem 8:* Consider a coded compressed sensing scenario, without successive interference cancellation, where all the parity bits are located in the trailing sub-blocks. The CCS algorithm can decode a message with vanishing error probability $P_e \to 0$, asymptotically in $K_a$, whenever the total number of parity bits is $P = (n + \delta - 1) \log_2 K_a$ for some fixed constant $\delta > 0$. Complexity orders for this allocation strategy appear in Table III. Constant $c_1$ from Table II is chosen such that $c_1 > 1$.

TABLE III

ORDER PERFORMANCE OF THE CCS ALGORITHM WHEN ALL THE PARITY BITS ARE LOCATED TOWARDS THE END OF CODEWORDS

|  | Regime (i) | Regime (ii) |
|---|---|---|
| $N$ | $\mathcal{O}(K_{\mathrm{a}} \log_2 K_{\mathrm{a}})$ | $\mathcal{O}(K_{\mathrm{a}}^2)$ |
| $C_{\mathrm{cs}}$ | $\mathcal{O}\left(K_{\mathrm{a}}^{c_1}\right)$ | $\mathcal{O}\left(\frac{K_{\mathrm{a}}^{c_1+1}}{\log_2 K_{\mathrm{a}}}\right)$ |
| $\mathbb{E}[C_{\mathrm{tree}}]$ | $\mathcal{O}\left(K_{\mathrm{a}}^{\alpha/c_1} \log_2 K_{\mathrm{a}}\right)$ | $\mathcal{O}\left(2^{K_{\mathrm{a}}/c_1} \log_2 K_{\mathrm{a}}\right)$ |
| $\mathbb{E}[C]$ | $\mathcal{O}\left(\max\{K_{\mathrm{a}}^{c_1}, K_{\mathrm{a}}^{\alpha/c_1} \log_2 K_{\mathrm{a}}\}\right)$ | $\mathcal{O}\left(2^{K_{\mathrm{a}}/c_1} \log_2 K_{\mathrm{a}}\right)$ |

TABLE IV

ORDER PERFORMANCE OF THE CCS ALGORITHM WHEN PARITY BITS ARE ASSIGNED UNIFORMLY ACROSS SUB-BLOCKS

|  | Regime (i) | Regime (ii) |
|---|---|---|
| $N$ | $\mathcal{O}(K_{\mathrm{a}} \log_2 K_{\mathrm{a}})$ | $\mathcal{O}(K_{\mathrm{a}}^2)$ |
| $C_{\mathrm{cs}}$ | $\mathcal{O}\left(K_{\mathrm{a}}^{c_1}\right)$ | $\mathcal{O}\left(\frac{K_{\mathrm{a}}^{c_1+1}}{\log_2 K_{\mathrm{a}}}\right)$ |
| $\mathbb{E}[C_{\mathrm{tree}}]$ | $\mathcal{O}(K_{\mathrm{a}} \log_2 K_{\mathrm{a}})$ | $\mathcal{O}(K_{\mathrm{a}}^2)$ |
| $\mathbb{E}[C]$ | $\mathcal{O}\left(K_{\mathrm{a}}^{c_1}\right)$ | $\mathcal{O}\left(\frac{K_{\mathrm{a}}^{c_1+1}}{\log_2 K_{\mathrm{a}}}\right)$ |

*Proof:* A proof is available in Appendix C. ∎

It can be seen from Theorem 8 and Table III that, in regime (ii), even though the average complexity scales sub-linearly in $2^{K_{\mathrm{a}}}$, it remains sizable as the gain in complexity only appears in the exponent, with $2^{K_{\mathrm{a}}/c_1}$. The decoding complexity for this scenario is high because the tree is not pruned during the initial stages of the process. The number of parity bits allocated to each node is zero for a constant fraction of the tree depth and, consequently, the tree grows exponentially until the first parity bit appears. To avoid such an exponential growth, the tree should be pruned during the initial stages via the early placement of parity bits.

### B. Parity Bits Equally Distributed Among Sub-Blocks

Again, we use $C_{\mathrm{cs}}$, $\mathbb{E}[C_{\mathrm{tree}}]$, $\mathbb{E}[C]$, and $N$ to denote the computational complexity of solving the $n$ CS sub-problems, the average tree decoding complexity, the average overall decoding complexity, and the total number of channel uses, respectively. Recall that having an equal number of parity-check bits per sub-block, except for the root block, limits complexity at the expense of performance. We present our findings below in the form of a theorem.

*Theorem 9:* Consider a coded compressed sensing scenario, without successive interference cancellation, where parity bits are assigned uniformly across sub-blocks, except for the root fragment. Suppose that the non-negative parameters $\alpha$, $c_1$, and $c_2$ in Table II are chosen such that

$$2 < \frac{\alpha - c_1}{c_2 - 1} < c_1 - 1 \tag{25}$$

in regime (i), and

$$3 < \frac{1}{c_2} < c_1 - 1 \tag{26}$$

in regime (ii). Then, the CCS algorithm can decode users with vanishing error probability, $P_e \to 0$, asymptotically in $K_{\mathrm{a}}$. Furthermore, the corresponding complexity orders for this allocation strategy appear in Table IV.

*Proof:* A proof is available in Appendix C. ∎

The ratio in (25) is related to the number of information bits per non-root sub-block, which is equal to $b = \frac{\alpha - c_1}{c_2 - 1} \log_2 K_{\mathrm{a}}$ in regime (i). These bounds ensure that all the information bits are accounted for, while also leaving enough slack for parity bits to grow rapidly. As an example, the requirements for regime (i) are met by $\alpha = 5.25$, $c_1 = 4$, $c_2 = 1.5$. A similar statement applies to the ratio in (26). The number of information bits per non-root sub-block in regime (ii) is

$b = \frac{1}{c_2} \log_2 K_{\mathrm{a}} - o(1)$. In this case, $1/c_2$ should be large enough to make room for all the information bits, while also leaving enough space for parity bits. For example, parameters $c_1 = 6$ and $c_2 = 0.25$ are admissible in regime (ii). Finally, in both cases, the lower bound guarantees that the conditions of Assumption 3 are met with probability approaching one as $K_{\mathrm{a}} \to \infty$.

### C. Comparison of Parity Allocations

It is interesting to note that the number of channel uses for reliable message recovery is of the same order when all the parity bits are located towards the end of the codewords and when the parity bits are equally distributed among all the non-root sub-blocks. This statement is valid for both regime (i) and regime (ii). The complexity order on the other hand is more severe for the scenario where all the parity bits are at the end. This suggests that, for large systems, it is beneficial to distribute parity bits across sub-blocks. The per step, multiplicative growth of the tree is $K_{\mathrm{a}}$. Reliable performance is achieved by bringing the expected number of surviving erroneous candidates very close to zero at every step. In the large scale setting, the tradeoff between performance and complexity appears to be favoring aggressive pruning early on, with parity bits being distributed equally among non-root sub-blocks. This parity allocation scheme is a particular case of what we observe for finite systems through numerical simulations. In the latter case, parity bits are assigned sparsely until the decoding tree reaches a certain expected girth, and they are assigned regularly thereafter. These findings are reported below.

## VI. SIMULATION RESULTS

In this section, we study the performance of the proposed scheme for realistic parameters attuned to practical application scenarios. We also provide a comparison with existing results found in the literature. We consider a system with $K_{\mathrm{a}} \in [25 : 300]$ active users, each having $B = 75$ bits of information to transmit. These bits are divided into $n = 11$ sub-blocks, each of length $J$; the specific value for $J$ depends on $K_{\mathrm{a}}$. Design attributes are summarized in Table V. In a manner akin to [5], we adopt sensing matrices that are constructed based on BCH codes. These matrices are employed for signaling in the compressed sensing sub-problems. More precisely, we select a subset $\mathcal{C}^0$ of codewords of size $|\mathcal{C}^0| = 2^J$ from the (2047,23) BCH codebook $\mathcal{C}$. This subset is created to possess

TABLE V

SUMMARY OF THE OPERATIONAL PARAMETERS USED IN SIMULATING THE
CCS ALGORITHM

| $K_a$ | 25 | 50 | 75 | 100 | 125 | 150 |
|---|---|---|---|---|---|---|
| $J$ | 14 | 14 | 14 | 14 | 14 | 15 |
| $\varepsilon_{\text{tree}}$ | 0.0025 | 0.0045 | 0.006 | 0.01 | 0.0125 | 0.0055 |
| $K_a$ | 175 | 200 | 225 | 250 | 275 | 300 |
| $J$ | 15 | 15 | 15 | 15 | 15 | 15 |
| $\varepsilon_{\text{tree}}$ | 0.0065 | 0.007 | 0.008 | 0.01 | 0.0125 | 0.0175 |

TABLE VI

THESE ARE THE RESULTS FROM THE OPTIMIZATION FRAMEWORK
DEVELOPED IN SECTION IV-C. ERROR PROBABILITY IS MINIMIZED
WHEN PARITY CHECK BITS ARE PUSHED TO END, WHEREAS
AVERAGE COMPUTATIONAL COMPLEXITY IS LEAST WHEN
EQUAL PARITY-CHECK BITS ARE ALLOCATED PER SUB-
BLOCK

| $\varepsilon_{\text{tree}}$ | $\mathbb{E}\big[\tilde{C}_{\text{tree}}\big]$ | Parity Length Vector |
|---|---|---|
| 0.006 | Infeasible | Infeasible |
| 0.0061930 | $3.2357 \times 10^{11}$ | [0 ,0, 0, 0, 15, 15, 15, 15, 15, 15] |
| 0.0061931 | 3357300 | [0, 3, 8, 8, 8, 8, 10, 15, 15, 15] |
| 0.0061932 | 1737000 | [0, 4, 8, 8, 8, 8, 9, 15, 15, 15] |
| 0.0061933 | 926990 | [0, 5, 8, 8, 8, 8, 8, 15, 15, 15] |
| 0.0061935 | 467060 | [1, 8, 8, 8, 8, 8, 8, 11, 15, 15] |
| 0.0062 | 79634 | [1, 8, 8, 8, 8, 8, 8, 11, 15, 15] |
| 0.007 | 7358 | [6, 8, 8, 8, 8, 8, 8, 8, 13, 15] |
| 0.008 | 6153 | [7, 8, 8, 8, 8, 8, 8, 8, 12, 15] |
| 0.02 | 5023 | [6, 8, 8, 9, 9, 9, 9, 9, 9, 14] |
| 0.04 | 4158 | [7, 8, 8, 9, 9, 9, 9, 9, 9, 13] |
| 0.6378 | 3066 | [9, 9, 9, 9, 9, 9, 9, 9, 9, 9] |

the following properties: $\underline{0} \in \mathcal{C}^0$; $\underline{c} \in \mathcal{C}^0$ implies $\underline{1} \oplus \underline{c} \in \mathcal{C} \backslash \mathcal{C}^0$; if $\underline{c}_1, \underline{c}_2 \in \mathcal{C}^0$ then $\underline{c}_1 \oplus \underline{c}_2 \in \mathcal{C}^0$. Above, $\underline{0}$ and $\underline{1}$ are the all-zero and all-one vectors, respectively. The $\oplus$ operator represents binary addition and, as such, $\underline{1} \oplus \underline{c}$ denotes the ones' complement of $\underline{c}$. We then choose sensing matrix $\mathbf{A}$ as $\mathbf{A} = [\underline{a}_0, \underline{a}_1, \ldots, \underline{a}_{2^J-1}]$ with dimension $2047 \times 2^J$, where $\underline{a}_j = \sqrt{E_s}(2\underline{c}_j - 1)$ and $\underline{c}_j \in \mathcal{C}^0$ for $j \in [0 : 2^J - 1]$. With this construction, the total number of channel uses is given by $N = 11 \times 2047 = 22,517$. The target error probability for the system is fixed at $\varepsilon = 0.05$.

We set list size $K$ for the NNLS CS sub-problems to $K = K_a + 10$. The computational complexity of the active-set method proposed by Lawson and Hanson [26] for NNLS is $\mathcal{O}(K^3 + Knp)$, where $K$ is the sparsity index. Parameters $n$ and $p$ are the numbers of rows and columns of the sensing matrix, respectively. For selected values $K_a \in [25 : 300]$, we solve the optimization problem (23) with CVX [27]. The solution then dictates the assignment of parity bits, as described in Section IV-C. Prescribed values for $\varepsilon_{\text{tree}}$ appear in Table V as a function of $K_a$. Parameters $B$ and $N$ are chosen such that the rate $\frac{B}{N} = \frac{75}{22,517}$ is approximately equal to the rate resulting from the choice of parameters $B = 100$ and $N = 30,000$ in [4], [5]. This enables a fair comparison between the schemes contained therein and the present approach.
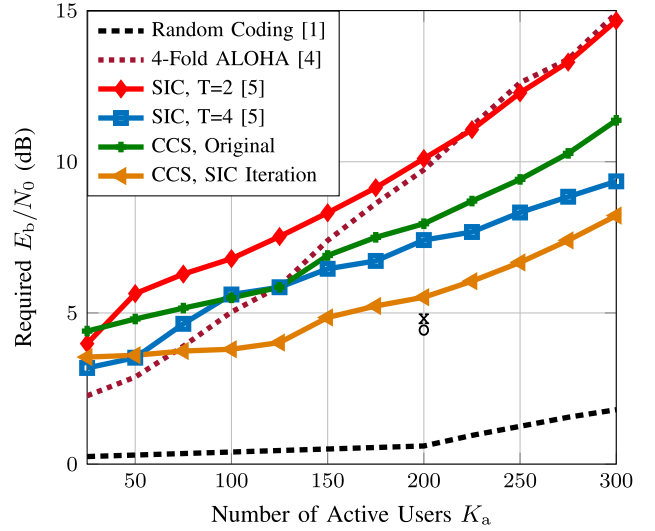


Fig. 7. This graph plots the minimum $E_b/N_0$ required to achieve $P_e \leq 0.05$ as a function of $K_a$, the number of active users. The proposed scheme and its enhanced version, with one iteration, have their own curves. The more demanding versions with 2 and 3 iterations are computed for a single operating point; they are represented by x and ○, respectively.

Table VI captures the tradeoff between performance and average computational cost as a function of the parity assignment vector for $K_a = 200$. These results are consistent with our treatment of the CCS algorithm. The early allocation of parity bits decreases complexity at the expense of performance. Suitable allocations tend to have a ramp-up phase where bits are allocated sparsely while the expected girth of the decoding tree grows, followed by a stable phase where the girth of the tree is maintained by a steady allocation of parity bits. Finally there is a heavy pruning phase where sub-blocks are filled almost exclusively with parity bits. Figure 7 displays the $E_b/N_0$ required to achieve a target error probability of 0.05 as a function of $K_a$ for various schemes. The bottom most curve corresponds to Polyanskiy's achievability bound [1] on the performance of a finite block length code for the communication problem at hand. The curves labelled $T = 2$, $T = 4$, and 4-fold ALOHA, which correspond to the performance curves in [5] and [4], assume the existence of a code for the $T$-user MAC channel which achieves the bound in [1]. It can be seen from Fig. 7 that our proposed approach with just one extended round of iteration outperforms previously published schemes for $K_a \in [75 : 300]$.

## VII. CONCLUSION

In this article, we propose a novel compressed sensing architecture for the unsourced random access problem. This scheme, called coded compressed sensing, innovatively combines techniques from compressed sensing and forward error correction to yield an algorithm with a manageable computational complexity. This conceptual structure, when applied to unsourced random access, offers significant performance improvements compared to older, low-complexity algorithms. After the publication of the conference version of this work [28], the tree code framework has been applied to design novel
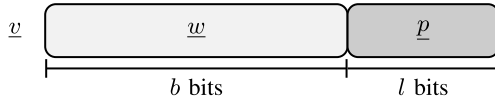
Fig. 8. Message $\underline{v}$ is obtained through systematic encoding. It is composed of $b$ information bits and $l$ parity-check bits.

random access schemes in [29] and [30]. In [29], a tree code is used in conjunction with a compressed sensing scheme based on sparse regression codes and approximate message passing. This scheme shows performance improvement over the proposed scheme in certain regimes. In [30], the tree code framework is used in conjunction with a Hadamard matrix based compressing scheme, which permits very low complexity decoding even for large number of users.

The CCS framework can be leveraged with a variety of sensing matrices, beyond those featured in this article. It can also serve as a conceptual blueprint for the design of algorithms in the context of very high-dimensional problems. In this sense, this technical approach may be relevant to many compressed sensing applications where the unknown sparse vector is very large. This may include, for instance, IoT neighbor discovery and the detection of heavy hitters in wired networks [10], [12], [22]. Still, the treatment of such problems would be a significant departure from the primary focus of this article and, as such, the aforementioned opportunities are left as potential avenues for future research.

## APPENDIX A
## PERFORMANCE OF TREE CODE

In this section, we characterize the performance of tree codes, as they pertain to coded compressive sensing. To do so, we briefly review relevant results related to random linear codes. We then extend these results to the desired setting. As we will see shortly, the exact analysis of the probability of failure under tree decoding, although tractable, is cumbersome, especially for deep trees. Nevertheless, a precise expression can be derived for performance characterization, and it is employed to compute probabilities of error for the parameters of interest.

### A. Random Linear Codes

To begin, we inspect the type of (random) linear codes we are interested in. Consider a binary information vector $\underline{w}$ of length $b$. This vector is systematically encoded using parity generator matrix $G$ of size $b \times l$. The resulting codeword, $\underline{v} = \underline{w}\underline{p}$, is then obtained by taking message $\underline{w}$ and appending parity vector $\underline{p}$ to it. Specifically, the parity bits are generated via linear equation

$$\underline{p} = \underline{w}G,$$

where operations are taken over binary field $\mathbb{F}_2$. Thus, $\underline{v}$ has length $b + l$, as depicted in Fig. 8.

Suppose that an alternate information vector $\underline{w}_\mathrm{r}$ is selected at random from $\{0,1\}^b$. This vector is encoded using the same generator matrix $G$, yielding concatenated vector $\underline{v}_\mathrm{r} = \underline{w}_\mathrm{r}\underline{p}_\mathrm{r}$

with $\underline{p}_\mathrm{r} = \underline{w}_\mathrm{r}G$. We wish to compute the probability that $\underline{v}$ and $\underline{v}_\mathrm{r}$ share a same parity sub-component, i.e., $\underline{p} = \underline{p}_\mathrm{r}$.

*Lemma 10:* Fix information vector $\underline{w}$ and parity generating matrix $G$. The probability that a randomly selected information vector $\underline{w}_\mathrm{r} \in \{0,1\}^b$ produces the same parity sub-component as $\underline{w}$ under $G$ is given by

$$\mathrm{Pr}\left(\underline{p} = \underline{p}_\mathrm{r}\right) = 2^{-\,\mathrm{rank}(G)}.$$

*Proof:* Information vector $\underline{w}_\mathrm{r}$ is drawn at random, uniformly over $\{0,1\}^b$. Codeword $\underline{v}_\mathrm{r}$ is then created by computing parity bits $\underline{p}_\mathrm{r} = \underline{w}_\mathrm{r}G$ and, subsequently, appending these bits to the information vector. This process yields codeword $\underline{v}_\mathrm{r} = \underline{w}_\mathrm{r}\underline{p}_\mathrm{r}$. The event $\{\underline{p} = \underline{p}_\mathrm{r}\}$ can therefore be expressed as

$$\{\underline{p} = \underline{p}_\mathrm{r}\} = \{\underline{w}G = \underline{w}_\mathrm{r}G\} = \{(\underline{w} + \underline{w}_\mathrm{r})G = \underline{0}\}$$
$$= \{\underline{w} + \underline{w}_\mathrm{r} \in \mathrm{nullspace}(G)\}.$$

Note that subtraction and addition are equivalent operations in $\mathbb{F}_2$. Moreover, vector $\underline{w} + \underline{w}_\mathrm{r}$ is uniformly distributed over $\{0,1\}^b$. Since the number of vectors in the nullspace of $G$ is $2^{\mathrm{nullity}(G)} = 2^{b-\mathrm{rank}(G)}$, it follows that

$$\mathrm{Pr}\left(\underline{p} = \underline{p}_\mathrm{r}\right) = \frac{2^{b-\mathrm{rank}(G)}}{2^b} = \frac{1}{2^{\mathrm{rank}(G)}}.$$

This completes the proof. ∎

The rank of parity generator matrix $G$ is of fundamental importance in assessing performance. In many scenarios, including our current treatment of unsourced multiple access, matrix $G$ is drawn at random from a large ensemble. In such cases, it becomes appropriate to compute the probability that an erroneous vector fulfills the parity constraints associated with $\underline{w}$.

*Lemma 11:* Fix erroneous vector $\underline{w}_\mathrm{e} \neq \underline{w}$. The entries of parity generator matrix $G \in \{0,1\}^{b \times l}$ are drawn at random from a uniform Bernoulli distribution, independently of one another. Under such circumstances, the probability of event $\{\underline{p} = \underline{p}_\mathrm{e}\}$ is given by

$$\mathrm{Pr}\left(\underline{p} = \underline{p}_\mathrm{e}\right) = 2^{-l}.$$

*Proof:* The event $\{\underline{p} = \underline{p}_\mathrm{e}\}$ is equivalent to the event $\{(\underline{w}+\underline{w}_\mathrm{e})G = \underline{0}\}$. Since $\underline{w} \neq \underline{w}_\mathrm{e}$, there exists at least one pair of vector entries, say at location $q$, such that $\underline{w}(q)+\underline{w}_\mathrm{e}(q) = 1$. Then,

$$(\underline{w} + \underline{w}_\mathrm{e})G = G[q,:] + \sum_{\ell \neq q}(\underline{w}(\ell) + \underline{w}_\mathrm{e}(\ell))G[\ell,:],$$

where $G[q,:]$ denotes the $q$th row of $G$. By constructions, $G[q,:]$ is a sequence of independent, uniform Bernoulli trials. Therefore, $(\underline{w} + \underline{w}_\mathrm{e})G$ also forms a sequence of independent, uniform Bernoulli bits and, hence, the probability that this sequence is the all-zero sequence is equal to $2^{-l}$. ∎

Conceptually, the results presented in this section form the cornerstones of our upcoming analysis. However, when codewords are fragmented and redundancy is added at different stages, assessing performance becomes more complicated. We initiate our treatment of this more elaborate scenario next.
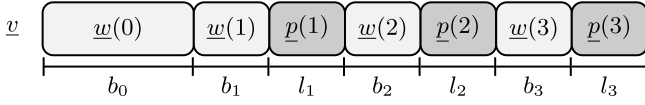
Fig. 9. The fragmented nature of tree coding leads to peculiarities in the performance analysis of this scheme. This situation arises, partly, from the fact that two different information sequences can share identical fragments.

### B. Fragmented Codewords

An important distinction that arises in coded compressive sensing when compared to standard random linear codes stems from the fragmented structure of tree codes depicted in Fig. 9. A second related factor ensues from the fact that erroneous candidates are formed by piecing together fragments from valid codewords. Typically, when studying error events under random linear coding, the starting point is an erroneous message of the form $\underline{w}_e \neq \underline{w}$, as in Lemma 11. However, the situation is more intricate for tree coding and meticulous accounting must be performed.

There are essentially three confounding factors in the analysis of erroneous messages in tree coding. First, several fragments within an erroneous candidate codeword may come from a same message. Such occurrences can significantly reduce the discriminating power of parity bits. Second, two different messages may have overlapping information fragments, as they only need to differ at one location overall. Thence, when comparing a valid codeword to an erroneous candidate, the two messages are necessarily distinct, yet some of their information fragments may be identical. Third, the loss of discriminating power from parity bits is correlated across fragments in certain cases, which exacerbates the probability of detection failure. We elaborate on these idiosyncrasies below.

Suppose that $\underline{w} = \underline{w}(0)\underline{w}(1)\cdots\underline{w}(n-1)$ is a partitioned information vector. Let

$$\underline{v} = \underline{w}(0)\underline{w}(1)\underline{p}(1)\cdots\underline{w}(n-1)\underline{p}(n-1) \qquad (27)$$

be the corresponding codeword produced by tree encoding. Since parity bits only operate on information bits contained in fragments that precede a slot, the generator matrix admits an upper triangular block structure of the form

$$G = \begin{bmatrix} G_{0,0} & G_{0,1} & G_{0,2} & \cdots \\ \mathbf{0} & G_{1,1} & G_{1,2} & \cdots \\ \mathbf{0} & \mathbf{0} & G_{2,2} & \cdots \\ \vdots & \vdots & \vdots & \ddots \end{bmatrix}.$$

The parity bits $\underline{p}(j)$ are then produced by column group $j$,

$$\underline{p}(j) = \sum_{\ell=0}^{j-1} \underline{w}(\ell)G_{\ell,j-1}.$$

These parity bits are subsequently injected into the codeword as in (27). We wish to capture the performance of the tree code when active blocks within $G$ are random, binary sub-matrices. That is, the entries of every sub-matrices are uniform Bernoulli trials, independent of one another and across blocks.

To examine the behavior of the tree code, consider a collection of codewords $\mathcal{L} = \{\underline{v}_1, \underline{v}_2, \ldots, \underline{v}_K\}$. Under tree coding, every codeword has the form

$$\underline{v}_i = \underline{w}_i(0)\underline{w}_i(1)\underline{p}_i(1)\cdots\underline{w}_i(n-1)\underline{p}_i(n-1)$$

where $\underline{p}_i(j) = \sum_{\ell=0}^{j-1} \underline{w}_i(\ell)G_{\ell,j-1}$. The collection of fragment candidates at stage 0 is

$$\mathcal{L}_0 = \{\underline{w}_i(0) : i = 1, \ldots, K\}.$$

Similarly, the list of candidate sub-blocks at stage $j$, where $j \in [1 : n-1]$, can be expressed as

$$\mathcal{L}_j = \{\underline{w}_i(j)\underline{p}_i(j) : i = 1, \ldots, K\}.$$

The goal of the tree decoder is to produce an unordered list that matches the original set of messages $\mathcal{M} = \{\underline{w}_1, \underline{w}_2, \ldots, \underline{w}_K\}$. We emphasize that valid messages whose sub-blocks appear on the corresponding lists invariably appear on the final CCS output of consistent codewords because they necessarily fulfill their own parity constraints. The probability of error is then dictated by instances where invalid codewords also meet their parity constraints. In such cases, the decoder fails to identify these extraneous candidates as erroneous and, therefore, they make it to the overall output list as well. As mentioned above, one of the challenges in analyzing error events is the fact that distinct candidate codewords are not guaranteed to have different fragments at any particular stage. That is, information fragments can appear more than once in $\mathcal{M}_j = \{\underline{w}_1(j), \underline{w}_2(j), \ldots, \underline{w}_K(j)\}$, possibly paired with different parity bits. This situation precludes the naive application of Lemma 11 in a divide-and-conquer fashion.

Consider the situation where the tree decoder seeks to validate codewords that start with root fragment $\underline{w}_{i_0}(0)$. For a given collection $\mathcal{L}$ of transmitted codewords, the list of candidate codewords visited during this portion of the tree decoding process is composed of elements of the form

$$\begin{aligned} \underline{v}_c &= \underline{v}_{i_0}(0)\underline{v}_{i_1}(1)\underline{v}_{i_2}(2)\cdots\underline{v}_{i_{n-1}}(n-1) \\ &= \underline{w}_{i_0}(0)\underline{w}_{i_1}(1)\underline{p}_{i_1}(1)\cdots\underline{w}_{i_{n-1}}(n-1)\underline{p}_{i_{n-1}}(n-1) \end{aligned} \qquad (28)$$

where $i_j \in [1 : K]$ for all slots $j \in [1 : n-1]$. The ability of the tree code to discriminate between valid and erroneous sequences hinges on the structure of the candidate vectors in (28) and their connection to parity generator matrix $G$. We turn our attention to the role of $G$ in identifying invalid codewords.

To survive stage 1, candidate vector $\underline{v}_c$ must fulfill condition $\underline{p}_{i_1}(1) = \underline{w}_{i_1}(0)G_{0,0} = \underline{w}_{i_0}(0)G_{0,0}$. Equivalently, we can write $(\underline{w}_{i_0}(0) + \underline{w}_{i_1}(0))G_{0,0} = \underline{0}$. To outlast the first two stages, a candidate vector must fulfill two conditions,

$$\underline{p}_{i_1}(1) = \underline{w}_{i_1}(0)G_{0,0} = \underline{w}_{i_0}(0)G_{0,0}$$

$$\begin{aligned} \underline{p}_{i_2}(2) &= (\underline{w}_{i_2}(0), \underline{w}_{i_2}(1)) \begin{bmatrix} G_{0,1} \\ G_{1,1} \end{bmatrix} \\ &= (\underline{w}_{i_0}(0), \underline{w}_{i_1}(1)) \begin{bmatrix} G_{0,1} \\ G_{1,1} \end{bmatrix}. \end{aligned}$$

This translates into linear equations

$$\left(\underline{w}_{i_0}(0) + \underline{w}_{i_1}(0)\right) G_{0,0} = \underline{0}$$

$$\left(\underline{w}_{i_0}(0) + \underline{w}_{i_2}(0), \underline{w}_{i_1}(1) + \underline{w}_{i_2}(1)\right) \begin{bmatrix} G_{0,1} \\ G_{1,1} \end{bmatrix} = \underline{0}.$$

In general, to survive the first $j$ stages, a candidate vector must meet the constraints

$$\sum_{\ell=0}^{q-1} \left(\underline{w}_{i_\ell}(\ell) + \underline{w}_{i_q}(\ell)\right) G_{\ell,q-1} = \underline{0} \qquad (29)$$

for $q = 1, \ldots, j$. Keeping these requirements in mind and factoring in the randomness in the problem, we can characterize the probability that an arbitrary candidate vector $\underline{v}_c$ survives the first $j$ stages by analyzing the number of nontrivial random linear equations it must satisfy. In addressing this question, it is meaningful to study the effective number of parity constraints this candidate sequence must meet on a per slot basis; that is, owing to the structure of a particular candidate codeword, some conditions may be fulfilled trivially. The insight afforded by Lemma 11 is that the $l_j$ parity bits associated with stage $j$ either act as a statistically discriminating sequence of independent Bernoulli samples, each with probability half, or they readily pass the parity check based on the combined vector in (29) vanishing. More specifically, the criterion for the former condition to apply is that at least one of the entries in

$$\left(\underline{w}_{i_0}(0) + \underline{w}_{i_j}(0), \ldots, \underline{w}_{i_{j-1}}(j-1) + \underline{w}_{i_j}(j-1)\right) \quad (30)$$

is not equal to zero. As mentioned above, there are two distinct phenomena that can drive individual entries in (30) to zero. Matching occurs in a block fashion whenever $i_\ell = i_j$ for some $\ell < j$. Under such circumstances, multiple portions of the candidate vector comes from valid codeword $\underline{w}_{i_j}$ itself and, consequently, $\underline{w}_{i_\ell}$ and $\underline{w}_{i_j}$ match at the corresponding locations. The second possibility arises from two distinct valid messages randomly sharing a common information fragment. At stage $j$, the probability that two independent sections match one another is given by

$$\Pr\left(\underline{w}_{i_\ell}(\ell) = \underline{w}_{i_j}(\ell)\Big| i_\ell \neq i_j\right) = 2^{-b_\ell},$$

where $\ell \in \{0, \ldots, j-1\}$. Not surprisingly, this probability is intrinsically linked to the number of information bits contained in the sub-block. We already know from Lemma 11 that there is a dichotomy between parity bits associated with column group $j$. A collection $\underline{p}(j)$ of parity bits is either statistically discriminating or, as a block, it becomes uninformative. The distinction between these two situations stems from the character of the input vectors in (30), whether they are identical or different. One subtlety related to the number of active parity bits for a particular candidate codeword can be seen by stacking the combined vectors,

$$\left(\underline{w}_{i_0}(0) + \underline{w}_{i_1}(0)\right)$$

$$\left(\underline{w}_{i_0}(0) + \underline{w}_{i_2}(0), \underline{w}_{i_1}(1) + \underline{w}_{i_2}(1)\right)$$

$$\vdots$$

$$\left(\underline{w}_{i_0}(0) + \underline{w}_{i_j}(0), \ldots, \underline{w}_{i_{j-1}}(j-1) + \underline{w}_{i_j}(j-1)\right).$$

Whenever a candidate codeword draws multiple fragments from a same valid message, the events where the combined vectors vanish and the corresponding parity bits lose their discriminatory power become more likely within a block. These events are also correlated across blocks. For example, if $i_2 = i_3$ and $\underline{w}_{i_0}(0) + \underline{w}_{i_2}(0) = \underline{0}$, then $\underline{w}_{i_0}(0) + \underline{w}_{i_3}(0) = \underline{0}$. Consequently, for certain index patterns, the probability that an erroneous candidate meets its parity constraints at different stages cannot be treated separately. Altogether, the probability that candidate codeword $\underline{v}_c$ is consistent with its own parity bits depends on its index sequence, which can be taken as visits to distinct levels over the progression of this candidate message.

Let index sequence $i_0, i_1, \ldots, i_{n-1}$ be given. We are interested in conditional probabilities of the form

$$\Pr(T = t | i_0, i_1, \ldots, i_{n-1})$$

where $T$ is the number of statistically discriminating parity bits. Since $i_j \in [1 : K]$, there are $K^n$ possible conditions. A careful examination of the problem and, in particular, of (29) reveals that this probability is permutation invariant. That is, the state labeling is irrelevant, only the order in which previously visited states are reentered matters. Mathematically, the structure of the problem enables us to write

$$\Pr(T = t | i_0, i_1, \ldots, i_{n-1})$$
$$= \Pr(T = t | \pi(i_0), \pi(i_1), \ldots, \pi(i_{n-1}))$$

for any permutation $\pi(\cdot)$ of the integers $1, \ldots, K$. When the number of active devices exceeds the slot count, this symmetric property reduces the number of possible cases from $K^n$ to the $n$th Bell number (OEIS A000110) given by the formula

$$B_n = \sum_{j=0}^{n} S_n^{(j)}.$$

Above, $S_n^{(j)}$ represents the Stirling numbers of the second kind (OEIS A008277), which can be computed recursively using the formula $S_n^{(j)} = S_{n-1}^{(j-1)} + j S_{n-1}^{(j)}$.

### C. Pattern Sequences

To simplify the problem and reduce the total number of cases to be considered, we leverage equivalence classes of erroneous patterns wherein every sequence within a class shares the same probability of going undetected. The structure of the equivalence classes is informed by $j$-patterns (OEIS A008277). For completeness, we include an appropriate definition below.

*Definition 12 (j-Pattern Sequences):* A $j$-pattern sequence is an integer sequence

$$\underline{s} = (\underline{s}(0), \underline{s}(1), \ldots, \underline{s}(j-1))$$

such that $\underline{s}(\ell) = \ell + 1$ or $\underline{s}(\ell) = \underline{s}(q)$ for some $q \in [0 : \ell - 1]$.

We denote the collection of all $j$-pattern sequences by $\mathcal{P}_j$, where $1 \leq j \leq n$. There is a single 1-pattern sequence, $\mathcal{P}_1 = \{(1)\}$. Similarly,

$$\mathcal{P}_2 = \{(1,1), (1,2)\}$$
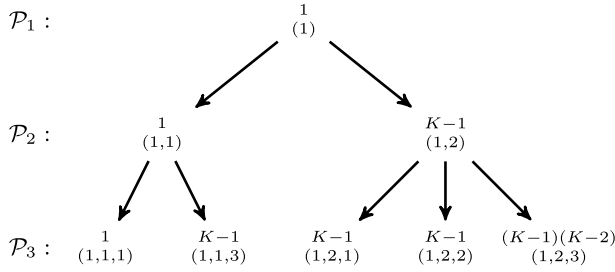$$\mathcal{P}_3 = \{(1,1,1), (1,1,3), (1,2,1), (1,2,2), (1,2,3)\}$$

Fig. 10. This illustration showcases $j$-pattern sequences for stages 0, 1, and 2, along with the sizes of their equivalence classes. The figure adopts the stacked format $\frac{n(\underline{s})}{\underline{s}}$.

indicate the collection of all 2-pattern and 3-pattern sequences, respectively. To be consistent with our established notation, we index the entries of $j$-pattern sequence $\underline{s}$ using integers $[0 : j-1]$. We emphasize that a $j$-pattern sequence can be created iteratively starting with a shorter $(j-1)$-pattern sequence $\underline{s}$ and appending to it either integer $j$ or entry $\underline{s}(k)$ for some $k < j-1$. With this construction, it can be verified that the total number of $j$-pattern sequences is equal to the $j$th Bell number (OEIS A000110); that is, $|\mathcal{P}_j| = B_j$.

Under this viewpoint, each $j$-pattern sequence $\underline{s} \in \mathcal{P}_j$ acts as a representative element to a collection of vectors of the form $(i_0, i_1, \ldots, i_{j-1})$. The mapping from an arbitrary integer sequences in $[1 : K]^n$ to the representative of its equivalence class can be defined recursively. The initial entry of the representative is $\underline{s}(0) = 1$, irrespective of $i_0$. Then, for any $q > 1$,

$$\underline{s}(q) = \begin{cases} q+1 & i_q \notin \{i_0, \ldots, i_{q-1}\} \\ \min\{\ell \in [1 : q-1] \mid i_\ell = i_q\} & \text{otherwise.} \end{cases}$$

For instance, any vector of the form $\{(i_0, i_1, i_2) : i_0 = i_1 = i_2\}$ maps to the 3-pattern sequence $(1, 1, 1)$. Likewise, any vector in $\{(i_0, i_1, i_2) : i_0 = i_1, i_1 \neq i_2\}$ map to $(1, 1, 3)$. The number of admissible integer sequences within an equivalence class is characterized below. Without loss of generality, we focus on a single rooted tree with $i_0 = 1$ to simplify our upcoming discussion.

*Lemma 13:* Let $\underline{s} \in \mathcal{P}_j$ denote a $j$-pattern sequence. The number of integer sequences $i_0, \ldots, i_{j-1}$ that map to $\underline{s}$ is given by

$$n(\underline{s}) = (K-1)(K-2)\cdots(K-(d(\underline{s})-1)), \qquad (31)$$

where $d(\underline{s})$ denotes the number of distinct integers in $\underline{s}$.

*Proof:* By assumption, we have $i_0 = 1$. For $\ell > 0$, integer $i_\ell$ is unambiguously determined by its predecessors whenever a previous entry is repeated in sequence $\underline{s}$. On the other hand, when $s(\ell) = \ell+1$, integer $i_\ell$ can be any element within $[1 : K]$ that has not appeared in the sequence thus far. Combining these observations with a straightforward application of the counting principle, we get (31). ∎

The total number of candidate paths in the tree at stage $j-1$ is $\sum_{\underline{s} \in \mathcal{P}_j} n(\underline{s}) = K^{j-1}$. Figure 10 shows $j$-pattern sequences and the cardinality of their equivalence classes for stages 0, 1, and 2 in the tree.

Our next task is to compute the probability that a representative from an equivalence class survives. We employ the notion of generating functions [31] to obtain the probability of survival. For every $j$-pattern sequence $\underline{s} \in \mathcal{P}_j$, we track the distribution of random variable $T(\underline{s})$, i.e., the number of effective parity-check bits for an element in that class. This is accomplished by evaluating the probability generating function (PGF) of $T(\underline{s})$, which takes the form

$$\Phi_{\underline{s}}(x) = \mathbb{E}\big[x^{T(\underline{s})}\big] = \sum_{t=0}^{l_1 + \cdots + l_{j-1}} \Pr(T(\underline{s}) = t)x^t. \qquad (32)$$

Above, $\Pr(T(\underline{s}) = t)$ is the probability that $t$ parity bits are statistically discriminating, for $j$-pattern sequence $\underline{s}$. The probability that a particular index sequence with representative class $\underline{s}$ survives at the end of the decoding process is then given by $\Phi_{\underline{s}}(0.5)$.

For example, the 3-pattern sequence $(1, 1, 1)$ has generating function $\Phi_{(1,1,1)}(x) = 1$ because it corresponds to a valid path in the tree. The parity bits for a valid sequence match the information bits by construction and, as such, this sequence invariably appears on the output list. The 3-pattern sequence $(1, 1, 3)$ represents states of the form $\{(i_0, i_1, i_2) : i_0 = i_1, i_1 \neq i_2\}$. From our previous discussion on fragmented codewords, we gather that the first $l_1$ parity bits in this candidate codeword are non-discriminating because the information bits for the first two fragments are drawn from a same message. On the other hand, the $l_2$ parity check bits in the second sub-block only become non-discriminating when $\underline{w}_{i_0}(0) = \underline{w}_{i_2}(0)$ and $\underline{w}_{i_0}(1) = \underline{w}_{i_2}(1)$. The conjunction of these events occurs with a probability $\frac{1}{2^{b_0+b_1}}$. Altogether, the generating function for this 3-pattern sequence is given by $\Phi_{(1,1,3)}(x) = \left(1 - \frac{1}{2^{b_0+b_1}}\right)x^{l_2} + \frac{1}{2^{b_0+b_1}}$. It can be verified in an analogous fashion that $\Phi_{(1,2,2)}(x) = \left(1 - \frac{1}{2^{b_0}}\right)x^{l_1+l_2} + \frac{1}{2^{b_0}}$.

Collecting the notions introduced thus far, we can define an aggregate generating function $\Phi(x)$ for the single rooted tree,

$$\Phi(x) = \sum_{\underline{s} \in \mathcal{P}_j} n(\underline{s})\Phi_{\underline{s}}(x). \qquad (33)$$

The expected number of surviving paths within the tree is then given by $\Phi(0.5)$. In the remainder of this section, we describe the computation of the generating functions for a given $j$-pattern sequence, $\Phi_{\underline{s}}(x)$. It is pertinent to note that $\Phi_{\underline{s}}(x)$ is sparse; the coefficient of $x^t$ can only be non-zero when

$$t \in \left\{\sum_{q \in S} l_q : S \subseteq [1 : j-1]\right\}.$$

Thus, it suffices to compute $\Pr(T(\underline{s}) = t)$ for values of $t$ within this set. We also point out that multiple subsets of $[1 : j-1]$ can contribute to a same coefficient. In other words, there can be distinct subsets, say $S$ and $S'$, such that $\sum_{q \in S} l_q = \sum_{q \in S'} l_q$. Altogether, the computation of $\Phi_{\underline{s}}(x)$ mandates inspecting the contribution of all $2^{j-1}$ subsets of $[1 : j-1]$.

*D. Statistically Discriminating Parity Bits*

Again, consider $j$-pattern sequence $\underline{s}$. We are interested in events where the parity bits associated with column groups

in $G$ are statistically discriminating. Recall that the $l_q$ parity-check bits in $\underline{p}(q)$ are generated via column group

$$
\begin{bmatrix}
G_{0,q-1} \\
\vdots \\
G_{q-1,q-1} \\
\mathbf{0}
\end{bmatrix},
$$

and they become collectively discriminating if and only if at least one entry in

$$
\left( \underline{w}_{\underline{s}(0)}(0) + \underline{w}_{\underline{s}(q)}(0), \ldots, \underline{w}_{\underline{s}(q-1)}(q-1) + \underline{w}_{\underline{s}(q)}(q-1), \underline{0} \right)
$$

is non-zero. Let $\mathcal{E}_{\underline{s},\mathcal{S}}$ be the event where the parity bits $\underline{p}(q)$ with $q \in \mathcal{S}$ are statistically discriminating, but those with indices in $\mathcal{S}^{\mathrm{c}} = [1 : n-1] \setminus \mathcal{S}$ are not. This somewhat intricate concept warrants an example. Suppose $\mathcal{S} = \{1, 3, 5\}$; then $\mathcal{E}_{\underline{s},\mathcal{S}}$ is the event where the parity-check bits $\underline{p}(1)$, $\underline{p}(3)$, and $\underline{p}(5)$ essentially become strings of random Bernoulli trials under the randomness in $G$, which must meet prescribed parity requirements for the candidate message to survive. Whereas the parity constraints for sub-blocks $[1 : n-1] \setminus \{1, 3, 5\}$ are fulfilled solely based on the $j$-pattern and/or message fragments matching one another, regardless of the realization of matrix $G$. Then, taking into account all the possible subsets of $[1 : n-1]$, we can rewrite (32) as

$$
\Phi_{\underline{s}}(x) = \sum_{\mathcal{S} \subseteq [1:n-1]} \Pr\left( \mathcal{E}_{\underline{s},\mathcal{S}} \right) x^{\sum_{q \in \mathcal{S}} l_q}. \tag{34}
$$

This expression invites a closer look at the summand $\Pr\left( \mathcal{E}_{\underline{s},\mathcal{S}} \right)$, which can be computed for every $\mathcal{S} \subseteq [1 : n-1]$ based on the randomness in the message sequences.

To better understand such events, we further decompose $\mathcal{E}_{\underline{s},\mathcal{S}}$ into events that occur within specific blocks. Let $\mathcal{A}_{q,\underline{s}}$ be the event in which parity-check bits of sub-block $q$ are non-discriminating for the $j$-pattern sequence $\underline{s}$. Based on our previous discussion on fragmented codewords, we have

$$
\mathcal{A}_{q,\underline{s}} = \bigcap_{\substack{j \in [0:q-1] \\ \underline{s}(q) \neq \underline{s}(j)}} \left\{ \underline{w}_{\underline{s}(q)}(j) = \underline{w}_{\underline{s}(j)}(j) \right\}. \tag{35}
$$

Then, event $\mathcal{E}_{\underline{s},\mathcal{S}}$ can be expressed as

$$
\mathcal{E}_{\underline{s},\mathcal{S}} = \left( \bigcap_{q \in \mathcal{S}^{\mathrm{c}}} \mathcal{A}_{q,\underline{s}} \right) \bigcap \left( \bigcap_{q \in \mathcal{S}} \mathcal{A}_{q,\underline{s}}^{\mathrm{c}} \right). \tag{36}
$$

We emphasize that the analysis of $\Pr\left( \mathcal{E}_{\underline{s},\mathcal{S}} \right)$ is cumbersome because the events $\mathcal{A}_{q,\underline{s}}$, $q \in [1 : j-1]$ are not independent from each other. Dependencies arise because of re-entries to previously visited levels, as captured by the $j$-pattern sequence $\underline{s}$. Nevertheless, it is meaningful to characterize the probability of building-block event $\mathcal{A}_{q,\underline{s}}$ before moving forward.

*Lemma 14:* Let $j$-pattern sequence $\underline{s}$ be fixed. The probability that the parity bits of sub-block $\underline{p}(q)$ are non-discriminating is

$$
\Pr\left( \mathcal{A}_{q,\underline{s}} \right) = g\left( \sum_{\substack{j \in [0:q-1] \\ \underline{s}(q) \neq \underline{s}(j)}} b_j \right),
$$

where $g(t) = 2^{-t}$.

*Proof:* We already know that, based on equivalence class $\underline{s}$, the contribution of block $G_{j,q-1}$ is lost at every location where $\underline{s}(j) = \underline{s}(q)$. As such, event $\mathcal{A}_{q,\underline{s}}$ corresponds to a situation where candidate message sequence $\left( \underline{w}_{\underline{s}(0)}(0), \ldots, \underline{w}_{\underline{s}(q-1)}(q-1) \right)$ also matches valid partial message $\left( \underline{w}_{\underline{s}(q)}(0), \ldots, \underline{w}_{\underline{s}(q)}(q-1) \right)$ at every locations where $\underline{s}(j) \neq \underline{s}(q)$. This renders the parity-check bits in $\underline{p}(q)$ non-discriminating. The probability of this event occurring, under fixed sequence $\underline{s}$, is given by

$$
\Pr\left( \mathcal{A}_{q,\underline{s}} \right) = \Pr\left( \bigcap_{\substack{j \in [0:q-1] \\ \underline{s}(q) \neq \underline{s}(j)}} \left\{ \underline{w}_{\underline{s}(q)}(j) = \underline{w}_{\underline{s}(j)}(j) \right\} \right)
$$

$$
= \prod_{\substack{j \in [0:q-1] \\ \underline{s}(q) \neq \underline{s}(j)}} \Pr\left( \underline{w}_{\underline{s}(q)}(j) = \underline{w}_{\underline{s}(j)}(j) \right)
$$

$$
= \prod_{\substack{j \in [0:q-1] \\ \underline{s}(q) \neq \underline{s}(j)}} g(b_j) = g\left( \sum_{\substack{j \in [0:q-1] \\ \underline{s}(q) \neq \underline{s}(j)}} b_j \right), \tag{37}
$$

where the second equality is justified through the independence of information bits across distinct messages. ∎

In essence, Lemma 14 reveals how the equivalence class $\underline{s}$ and the randomness in information bits across messages determine the discriminatory power of $\underline{p}(q)$. Ultimately, the ability to detect an erroneous codeword through $\underline{p}(q)$ also depends on the randomness in $G$. Under the right conditions, there is a block activation of the $l_q$ bits; otherwise, they collectively become non-discriminating. At this point, we turn to the challenge of computing the joint probability that parity-check bits across blocks are discriminating. To account for dependencies among column groups, we express $\Pr\left( \mathcal{E}_{\underline{s},\mathcal{S}} \right)$ using conditional probability,

$$
\Pr\left( \mathcal{E}_{\underline{s},\mathcal{S}} \right) = \Pr\left( \bigcap_{q \in \mathcal{S}} \mathcal{A}_{q,\underline{s}}^{\mathrm{c}} \middle| \bigcap_{q \in \mathcal{S}^{\mathrm{c}}} \mathcal{A}_{q,\underline{s}} \right) \Pr\left( \bigcap_{q \in \mathcal{S}^{\mathrm{c}}} \mathcal{A}_{q,\underline{s}} \right). \tag{38}
$$

We begin our treatment of (38) by examining the rightmost term. We can apply the chain rule of probability to this expression, which yields

$$
\Pr\left( \bigcap_{q \in \mathcal{S}^{\mathrm{c}}} \mathcal{A}_{q,\underline{s}} \right) = \prod_{q \in \mathcal{S}^{\mathrm{c}}} \Pr\left( \mathcal{A}_{q,\underline{s}} \middle| \bigcap_{\ell \in \mathcal{S}^{\mathrm{c}} \cap [1:q-1]} \mathcal{A}_{\ell,\underline{s}} \right). \tag{39}
$$

Inspecting (35), we gather that events of the form $\mathcal{A}_{q,\underline{s}}$ are subject to a partial order through set inclusion. In particular, suppose that $\underline{s}(q) = \underline{s}(\ell)$ with $\ell < q$, then $\mathcal{A}_{q,\underline{s}} \subset \mathcal{A}_{\ell,\underline{s}}$. This property immediately implies that the most constraining condition within this branch of the partial order is $\mathcal{A}_{p,\underline{s}}$ where $p = \max\left\{ \ell \in [1 : q-1] \middle| \underline{s}(q) = \underline{s}(\ell) \right\}$. On the other hand, when $\underline{s}(q) \neq \underline{s}(\ell)$, the events $\mathcal{A}_{q,\underline{s}}$ and $\mathcal{A}_{\ell,\underline{s}}$ are independent. This follows from the fact that, in this latter case, the conditions in (35) operate on different pairs of fragments altogether.

The fact that information bits are selected independently across messages is therefore enough to ensure independence. While the situation is slightly more involved when accounting for $\mathcal{S}$, the concepts are essentially identical. These observations invite the following definitions,

$$
\begin{aligned}
\underline{\mathcal{G}}_{q,\mathcal{S},\underline{s}} &= [1:q-1] \cap \mathcal{S}^{\mathrm{c}} \\
&\quad \cap \{k \in [1:j-1] : \underline{s}(k) \neq \underline{s}(q)\} \quad (40) \\
\underline{\mathcal{Q}}_{q,\mathcal{S},\underline{s}} &= [1:q-1] \cap \mathcal{S}^{\mathrm{c}} \\
&\quad \cap \{k \in [1:j-1] : \underline{s}(k) = \underline{s}(q)\}. \quad (41)
\end{aligned}
$$

In words, $\underline{\mathcal{G}}_{q,\mathcal{S},\underline{s}}$ contains the identification numbers of the sub-blocks that precede sub-block $q$ (excluding sub-block 0), feature a different level than $\underline{s}(q)$, and are not included in $\mathcal{S}$. Likewise, $\underline{\mathcal{Q}}_{q,\mathcal{S},\underline{s}}$ contains the numbers of the sub-blocks that precede sub-block $q$ (excluding sub-block 0), possess the same level as $\underline{s}(q)$, and are not included in $\mathcal{S}$. Based on this notation, we formalize the above discussion into a lemma.

*Lemma 15:* Consider equivalence class $\underline{s}$ along with set $\mathcal{S} \subset [1:n-1]$. Then, when the conditional event has positive probability, we get

$$
\begin{aligned}
&\Pr\left(\mathcal{A}_{q,\underline{s}} \,\middle|\, \bigcap_{\ell \in \mathcal{S}^{\mathrm{c}} \cap [1:q-1]} \mathcal{A}_{\ell,\underline{s}}\right) \\
&= g\left(\sum_{\substack{j \in [0:q-1] \\ \underline{s}(q) \neq \underline{s}(j)}} b_j - \sum_{\substack{k \in [0:p-1] \\ \underline{s}(p) \neq \underline{s}(k)}} b_k\right), \quad (42)
\end{aligned}
$$

where $p = \max \underline{\mathcal{Q}}_{q,\mathcal{S},\underline{s}}$ and $g(t) = 2^{-t}$.

*Proof:* Taking into consideration the independence structure identified above, we immediately get

$$
\begin{aligned}
&\Pr\left(\mathcal{A}_{q,\underline{s}} \,\middle|\, \bigcap_{\ell \in \mathcal{S}^{\mathrm{c}} \cap [1:q-1]} \mathcal{A}_{\ell,\underline{s}}\right) \\
&= \Pr\left(\mathcal{A}_{q,\underline{s}} \,\middle|\, \bigcap_{\ell \in \underline{\mathcal{Q}}_{q,\mathcal{S},\underline{s}}} \mathcal{A}_{\ell,\underline{s}}, \bigcap_{\ell \in \underline{\mathcal{G}}_{q,\mathcal{S},\underline{s}}} \mathcal{A}_{\ell,\underline{s}}\right) \\
&= \Pr\left(\mathcal{A}_{q,\underline{s}} \,\middle|\, \bigcap_{\ell \in \underline{\mathcal{Q}}_{q,\mathcal{S},\underline{s}}} \mathcal{A}_{\ell,\underline{s}}\right). \quad (43)
\end{aligned}
$$

We know that events of the form $\mathcal{A}_{q,\underline{s}}$ form a partial order through set inclusion. Moreover, $\mathcal{A}_{k,\underline{s}} \subset \mathcal{A}_{j,\underline{s}}$ whenever $j < k$ and $\underline{s}(j) = \underline{s}(k)$. This leads to the following two equalities:

$$
\bigcap_{k \in \underline{\mathcal{Q}}_{q,\mathcal{S},\underline{s}}} \mathcal{A}_{k,\underline{s}} = \mathcal{A}_{p,\underline{s}}
$$

and $\mathcal{A}_{q,\underline{s}} \cap \mathcal{A}_{p,\underline{s}} = \mathcal{A}_{q,\underline{s}}$, where $p = \max \underline{\mathcal{Q}}_{q,\mathcal{S},\underline{s}}$. Substituting these expressions into the conditional probability of (43),

we get

$$
\begin{aligned}
\Pr\left(\mathcal{A}_{q,\underline{s}} \,\middle|\, \bigcap_{\ell \in \underline{\mathcal{Q}}_{q,\mathcal{S},\underline{s}}} \mathcal{A}_{\ell,\underline{s}}\right) &= \Pr\left(\mathcal{A}_{q,\underline{s}} \,\middle|\, \mathcal{A}_{p,\underline{s}}\right) \\
&= \frac{\Pr\left(\mathcal{A}_{q,\underline{s}} \cap \mathcal{A}_{p,\underline{s}}\right)}{\Pr\left(\mathcal{A}_{p,\underline{s}}\right)} = \frac{\Pr\left(\mathcal{A}_{q,\underline{s}}\right)}{\Pr\left(\mathcal{A}_{p,\underline{s}}\right)} \\
&= g\left(\sum_{\substack{j \in [0:q-1] \\ \underline{s}(q) \neq \underline{s}(j)}} b_j - \sum_{\substack{k \in [0:p-1] \\ \underline{s}(p) \neq \underline{s}(k)}} b_k\right). \quad (44)
\end{aligned}
$$

The last inequality in (44) is obtained by applying Lemma 14 and using straightforward properties of the function $g(t) = 2^{-t}$. ∎

Together Lemma 15 and (39) offer a means to compute the rightmost term in (38). We then turn to the leftmost term on the right-hand-side of (38). To facilitate the description of this slightly more involved expression, we need to expand special cases beyond (40) and (41). Specifically, for every $q \in [1 : n-1]$, we introduce the following additional cases:

$$
\begin{aligned}
\overline{\mathcal{G}}_{q,\mathcal{S},\underline{s}} &= [q+1:n-1] \cap \mathcal{S}^{\mathrm{c}} \\
&\quad \cap \{k \in [1:n-1] : \underline{s}(k) \neq \underline{s}(q)\} \\
\overline{\mathcal{Q}}_{q,\mathcal{S},\underline{s}} &= [q+1:n-1] \cap \mathcal{S}^{\mathrm{c}} \\
&\quad \cap \{k \in [1:n-1] : \underline{s}(k) = \underline{s}(q)\} \\
\tilde{\underline{\mathcal{G}}}_{q,\mathcal{S},\underline{s}} &= [1:q-1] \cap \mathcal{S} \cap \{k \in [1:n-1] : \underline{s}(k) \neq \underline{s}(q)\} \\
\tilde{\underline{\mathcal{Q}}}_{q,\mathcal{S},\underline{s}} &= [1:q-1] \cap \mathcal{S} \cap \{k \in [1:n-1] : \underline{s}(k) = \underline{s}(q)\}.
\end{aligned}
$$

Above, $\overline{\mathcal{G}}_{q,\mathcal{S},\underline{s}}$ contains the numbers of the sub-blocks that trail sub-block $q$, feature a different level than $\underline{s}(q)$, and are not included in $\mathcal{S}$. The set $\overline{\mathcal{Q}}_{q,\mathcal{S},\underline{s}}$ contains the numbers of the sub-blocks that trail sub-block $q$, possess the same level as $\underline{s}(q)$, and are not included in $\mathcal{S}$. The last two conditions, $\tilde{\underline{\mathcal{G}}}_{q,\mathcal{S},\underline{s}}$ and $\tilde{\underline{\mathcal{Q}}}_{q,\mathcal{S},\underline{s}}$ are analogous to $\underline{\mathcal{G}}_{q,\mathcal{S},\underline{s}}$ and $\underline{\mathcal{Q}}_{q,\mathcal{S},\underline{s}}$, albeit they contain numbers belonging to $\mathcal{S}$ rather than its complement.

The strategy to tackle

$$
\Pr\left(\bigcap_{q \in \mathcal{S}} \mathcal{A}_{q,\underline{s}}^{\mathrm{c}} \,\middle|\, \bigcap_{\ell \in \mathcal{S}^{\mathrm{c}}} \mathcal{A}_{\ell,\underline{s}}\right) \quad (45)
$$

is similar to our previous step. We apply the chain rule of probability and analyze the relation between events. The expanded version of (45) can be written as

$$
\prod_{q \in \mathcal{S}} \Pr\left(\mathcal{A}_{q,\underline{s}}^{\mathrm{c}} \,\middle|\, \bigcap_{\ell \in \mathcal{S}^{\mathrm{c}}} \mathcal{A}_{\ell,\underline{s}}, \bigcap_{k \in \mathcal{S} \cap [1:q-1]} \mathcal{A}_{k,\underline{s}}^{\mathrm{c}}\right). \quad (46)
$$

We can further partition the conditional events via

$$
\mathcal{S}^{\mathrm{c}} = \underline{\mathcal{G}}_{q,\mathcal{S},\underline{s}} \uplus \underline{\mathcal{Q}}_{q,\mathcal{S},\underline{s}} \uplus \overline{\mathcal{G}}_{q,\mathcal{S},\underline{s}} \uplus \overline{\mathcal{Q}}_{q,\mathcal{S},\underline{s}} \quad (47)
$$

$$
\mathcal{S} \cap [1:q-1] = \tilde{\underline{\mathcal{G}}}_{q,\mathcal{S},\underline{s}} \uplus \tilde{\underline{\mathcal{Q}}}_{q,\mathcal{S},\underline{s}}. \quad (48)
$$

The event $\mathcal{A}_{q,\underline{s}}^{\mathrm{c}}$ in which parity-check bits of sub-block $q$ are discriminating; together with what occurs at level $\underline{s}(q)$, namely $\bigcap_{\ell \in \underline{\mathcal{Q}}_{q,\mathcal{S},\underline{s}} \uplus \overline{\mathcal{Q}}_{q,\mathcal{S},\underline{s}}} \mathcal{A}_{\ell,\underline{s}}$ and $\bigcap_{k \in \tilde{\underline{\mathcal{Q}}}_{q,\mathcal{S},\underline{s}}} \mathcal{A}_{k,\underline{s}}^{\mathrm{c}}$, is independent of sub-blocks associated with other levels, namely

$\bigcap_{\ell \in \underline{\mathcal{G}}_{q,\mathcal{S},\underline{s}} \uplus \overline{\mathcal{G}}_{q,\mathcal{S},\underline{s}}} \mathcal{A}_{\ell,\underline{s}}$ and $\bigcap_{k \in \tilde{\underline{\mathcal{G}}}_{q,\mathcal{S},\underline{s}}} \mathcal{A}_{k,\underline{s}}^{\mathrm{c}}$. As before, this property follows from the fact that, at levels other than $\underline{s}(q)$, the conditions in (35) operate on different pairs of fragments altogether. Since information bits are selected independently across messages, this is enough to ensure independence between the groupings above. Thus, (46) reduces to (49), shown at the bottom of the next page. The probability expression in (49) is in a form reminiscent of Lemma 15. At this stage, we proceed by breaking the conditional expression above into special cases.

*a) Case I – ($\overline{\mathcal{Q}}_{q,\mathcal{S},\underline{s}} \neq \emptyset$):* If this set is not empty, then there exists an element $\ell \in \overline{\mathcal{Q}}_{q,\mathcal{S},\underline{s}}$, $\ell > q$, such that the parity bits generated by column group $\ell$ are non-discriminating. This implies that vector $\left(\underline{w}_{\underline{s}(0)}(0) + \underline{w}_{\underline{s}(\ell)}(0), \ldots, \underline{w}_{\underline{s}(\ell-1)}(\ell-1) + \underline{w}_{\underline{s}(\ell)}(\ell-1)\right) = \underline{0}$. Since $\underline{s}(\ell) = \underline{s}(q)$, we deduce that sub-vector $\left(\underline{w}_{\underline{s}(0)}(0) + \underline{w}_{\underline{s}(q)}(0), \ldots, \underline{w}_{\underline{s}(q-1)}(q-1) + \underline{w}_{\underline{s}(q)}(q-1)\right) = \underline{0}$. Thus, the parity-check bits generated by column group $q$ are non-discriminating and

$$\Pr\left(\mathcal{A}_{q,\underline{s}} \middle| \bigcap_{\ell \in \underline{\mathcal{Q}}_{q,\mathcal{S},\underline{s}}} \mathcal{A}_{\ell,\underline{s}}, \bigcap_{\ell \in \overline{\mathcal{Q}}_{q,\mathcal{S},\underline{s}}} \mathcal{A}_{\ell,\underline{s}}, \bigcap_{k \in \tilde{\underline{\mathcal{Q}}}_{q,\mathcal{S},\underline{s}}} \mathcal{A}_{k,\underline{s}}^{\mathrm{c}}\right) = 1. \tag{50}$$

*b) Case II – ($\tilde{\underline{\mathcal{Q}}}_{q,\mathcal{S},\underline{s}} \neq \emptyset$):* If this set in not empty, then there exists an element $k \in \tilde{\underline{\mathcal{Q}}}_{q,\mathcal{S},\underline{s}}$, $k < q$, such that the parity bits generated by column group $k$ are discriminating. In other words, vector $\left(\underline{w}_{\underline{s}(0)}(0) + \underline{w}_{\underline{s}(k)}(0), \ldots, \underline{w}_{\underline{s}(k-1)}(k-1) + \underline{w}_{\underline{s}(k)}(k-1)\right)$ is non-zero. Yet, this vector appears as a sub-component of $\left(\underline{w}_{\underline{s}(0)}(0) + \underline{w}_{\underline{s}(q)}(0), \ldots, \underline{w}_{\underline{s}(q-1)}(q-1) + \underline{w}_{\underline{s}(q)}(q-1)\right)$ and, consequently, the latter vector must also be a non-zero vector. By Lemma 11, this implies that the parity-check bits associated with column group $q$ are discriminatory and

$$\Pr\left(\mathcal{A}_{q,\underline{s}} \middle| \bigcap_{\ell \in \underline{\mathcal{Q}}_{q,\mathcal{S},\underline{s}}} \mathcal{A}_{\ell,\underline{s}}, \bigcap_{\ell \in \overline{\mathcal{Q}}_{q,\mathcal{S},\underline{s}}} \mathcal{A}_{\ell,\underline{s}}, \bigcap_{k \in \tilde{\underline{\mathcal{Q}}}_{q,\mathcal{S},\underline{s}}} \mathcal{A}_{k,\underline{s}}^{\mathrm{c}}\right) = 0. \tag{51}$$

*c) Case III – ($\overline{\mathcal{Q}}_{q,\mathcal{S},\underline{s}} = \emptyset$ and $\tilde{\underline{\mathcal{Q}}}_{q,\mathcal{S},\underline{s}} = \emptyset$):* In this case, we get

$$\Pr\left(\mathcal{A}_{q,\underline{s}} \middle| \bigcap_{\ell \in \underline{\mathcal{Q}}_{q,\mathcal{S},\underline{s}}} \mathcal{A}_{\ell,\underline{s}}, \bigcap_{\ell \in \overline{\mathcal{Q}}_{q,\mathcal{S},\underline{s}}} \mathcal{A}_{\ell,\underline{s}}, \bigcap_{k \in \tilde{\underline{\mathcal{Q}}}_{q,\mathcal{S},\underline{s}}} \mathcal{A}_{k,\underline{s}}^{\mathrm{c}}\right)$$
$$= \Pr\left(\mathcal{A}_{q,\underline{s}} \middle| \bigcap_{\ell \in \underline{\mathcal{Q}}_{q,\mathcal{S},\underline{s}}} \mathcal{A}_{\ell,\underline{s}}\right) \tag{52}$$

and we can apply Lemma 15, which gives us a means to compute the desired probability.

*Remark 16:* The fourth case where both $\overline{\mathcal{Q}}_{q,\mathcal{S},\underline{s}} \neq \emptyset$ and $\tilde{\underline{\mathcal{Q}}}_{q,\mathcal{S},\underline{s}} \neq \emptyset$ cannot occur and need not be considered. The first condition prevents $\underline{p}(q)$ from being statistically discriminating, whereas the second condition forces these parity-check bits to be discriminating. In such scenarios, $\Pr\left(\mathcal{E}_{\underline{s},\mathcal{S}}\right)$ does not admit

the conditional form of (38), but it can readily be identified as having probability zero.

We collect the results discussed above and summarize our findings in the form of a proposition.

*Proposition 17:* Consider equivalence class $\underline{s}$ and set $\mathcal{S} \subset [1 : n-1]$. The probability that the parity-check bits $\underline{p}(q)$ with $q \in \mathcal{S}$ are statistically discriminating, and those with indices in $\mathcal{S}^{\mathrm{c}} = [1 : n-1] \setminus \mathcal{S}$ are non-discriminating is given by (53), shown at the bottom of the next page, where $\mathbb{1}_{\{\cdot\}}$ is the set indicator function, $p = \max \underline{\mathcal{Q}}_{q,\mathcal{S},\underline{s}}$ and $g(t) = 2^{-t}$.

This proposition, together with (33) and (34), provides an algorithmic procedure to compute the expected number of erroneous paths that survive the decoding process. While the proposition is stated for the full length of the transmission process, the reader will notice that the result can also be applied at any stage of the decoding process. Hence, the expected number of erroneous surviving paths at level $j-1$ is given by

$$\mathbb{E}[L_{j-1}] = \sum_{\underline{s} \in \mathcal{P}_j} n(\underline{s}) \Phi_{\underline{s}} \left(\frac{1}{2}\right) - 1, \quad j \in [1 : n].$$

Our expression for the expected number of surviving candidate codewords at the end of the tree decoding process is convoluted. Unfortunately, there is no apparent structure, beyond the presence of equivalence classes, that enable a reduction in computational complexity. It is pertinent to mention that the Bell numbers grow rapidly and, as such, the computation of the expected number of surviving paths through the tree can be challenging. Nevertheless, exact numbers can be obtained for the parameters we are interested in.

## APPENDIX B
### APPROXIMATE PERFORMANCE OF TREE CODE

This appendix contains proofs for some of the results contained in Section IV-B. Recall that the approximate analysis for tree coding is performed under Assumption 3. That is, the collection of messages $\mathcal{M}$ is such that, whenever $i \neq k$,

$$\underline{w}_i(j) \neq \underline{w}_k(j)$$

for all $j \in [0 : n-1]$ with $b_j > 0$.

### A. Proof of Proposition 4

We begin the proof with the simplest scenario. Consider erroneous partial vector $\underline{w}_{i_0}(0)\underline{w}_{i_1}(1)$ where $i_0 \neq i_1$. Under Assumption 3, we know that $\underline{w}_{i_0}(0) \neq \underline{w}_{i_1}(0)$. Then, by Lemma 11, the probability that $p_{i_0}(1) = p_{i_1}(1)$ is given by

$$\Pr\left(p_{i_0}(1) = p_{i_1}(1)\right) = 2^{-l_1}.$$

That is, the probability that an erroneous vector survives stage 1 is equal to $2^{-l_1}$. Moving forward, we can extend this analysis to subsequent stages. Consider erroneous partial vector $\underline{w}_{i_0}(0)\underline{w}_{i_1}(1) \cdots \underline{w}_{i_j}(j)$. Since this is not a valid message, there exists $q < j$ such that $i_q \neq i_j$. Under Assumption 3, we then have $\underline{w}_{i_q}(q) \neq \underline{w}_{i_j}(q)$. Again, by Lemma 11, we gather that

$$\Pr\left(\sum_{\ell=0}^{j-1} \underline{w}_{i_\ell}(\ell) G_{\ell,j-1} = \underline{p}_{i_j}(j)\right) = p_j, \tag{54}$$

where $p_j = 2^{-l_j}$. We note, briefly, that the event in (54) is equivalent to the parity condition found in (29), which is the requirement for a candidate vector to pass stage $j$.

Suppose $j \geq 2$ and assume $\tilde{L}_{j-1}$ is given. In view of (54), $\tilde{L}_j$ can be interpreted as the sum of $K(\tilde{L}_{j-1}+1)-1$ Bernoulli random variables, each with parameter $p_j$. Consequently, the expected value $\mathbb{E}[\tilde{L}_j]$ admits the recursive form

$$
\begin{aligned}
\mathbb{E}[\tilde{L}_j] &= \mathbb{E}\left[\mathbb{E}[\tilde{L}_j \mid \tilde{L}_{j-1}]\right] \\
&= \mathbb{E}\left[\left(\left(\tilde{L}_{j-1}+1\right)K-1\right)p_j\right] \\
&= p_j K \mathbb{E}[\tilde{L}_{j-1}] + p_j(K-1).
\end{aligned}
\tag{55}
$$

The above equation can be solved recursively using initial condition $\mathbb{E}[\tilde{L}_1] = (K-1)p_1$. This yields the closed-form expression that appears in (9).

## APPENDIX C
## ASYMPTOTIC PERFORMANCE ANALYSIS

This section contains demonstrations for the two main results contained in Section V, which focuses on the asymptotic analysis of CCS.

### A. Proof of Theorem 8

For mathematical convenience, we assume that the total number of parity bits $P$ is an integer multiple of the length of coded sub-blocks $J$. Since all the parity bits are located towards the end of the codewords, the final $P/J$ sub-blocks only contain parity bits, and the initial $n - P/J$ sub-blocks are composed of information bits. In other words, the number of parity bits in sub-block $j$ is given by

$$
l_j = \begin{cases} 0 & j < n - P/J \\ J & j \geq n - P/J. \end{cases}
\tag{56}
$$

For the scenario of interest, Assumption 3 can be stated as follows. The collection of messages decoded at the destination is such that, for any $i \neq k$,

$$
\underline{w}_i(j) \neq \underline{w}_k(j)
$$

for all $j \in [0 : n - 1 - P/J]$. This is due to the fact that $b_j = 0$ for all $j \in [n - P/J : n - 1]$. Also, all the sub-blocks

$j \in [0 : n - 1 - P/J]$ contain only information bits and no parity checks.

The reader may note that the compressed sensor returns only the collection of sub-blocks transmitted by active users in a slot. This implies that even if two or more users pick the same column from the sensing matrix and transmit it, the CS decoder only returns the index once. Since these sub-blocks are composed solely of information bits, we have $\underline{w}_i(j) \neq \underline{w}_k(j)$ for all $j \in [0 : n - 1 - P/J]$. In other words, with this parity-check profile, Assumption 3 holds with probability one and, consequently, all the parity bits are statistically discriminating. This situation is akin to the conditions of Lemma 11 in Appendix A-A. Thus, applying tree decoding in parallel, Proposition 4 and Proposition 5 become exact rather than an approximate characterization. Leveraging these findings, we derive conditions on $P$ for which $\mathbb{E}[L_{n-1}]$ decays to zero in the limit as $K_a \to \infty$. This immediately results in a vanishing tree decoding error probability because

$$
p_{\text{tree}} = \Pr(L_{n-1} \geq 1) \leq \mathbb{E}[L_{n-1}]
$$

by the Markov inequality. The quantity $\prod_{\ell=q}^{n-1} p_\ell$ appearing in (9) can be computed using (56) as

$$
\prod_{\ell=q}^{n-1} p_\ell = \begin{cases} \left(\frac{1}{2^J}\right)^{P/J} = \frac{1}{2^P} & q \leq n - P/J \\ \left(\frac{1}{2^J}\right)^{n-q} = \frac{1}{2^{J(n-q)}} & q > n - P/J. \end{cases}
$$

Substituting the above expression into (9), we can upper bound the expected value $\mathbb{E}[L_{n-1}]$ by

$$
\begin{aligned}
&\mathbb{E}[L_{n-1}] \\
&\leq \sum_{j=1}^{n-\frac{P}{J}} \frac{K_a^{n-1-j}(K_a-1)}{2^P} + \sum_{j=n-\frac{P}{J}+1}^{n-1} \frac{K_a^{n-1-j}(K_a-1)}{2^{J(n-j)}} \\
&= \frac{K_a^{P/J-1}}{2^P}\left[K^{n-P/J}-1\right] + \frac{K_a-1}{2^J}\left[\frac{1-\left(\frac{K_a}{2^J}\right)^{P/J-1}}{1-\frac{K_a}{2^J}}\right] \\
&\leq \frac{K_a^{n-1}}{2^P} + \frac{K_a}{2^J - K_a} \\
&\leq \frac{2^{(n-1)\log_2 K_a}}{2^P} + \frac{1}{K_a^{c_1-1}-1}.
\end{aligned}
$$

$$
\prod_{q \in \mathcal{S}} \Pr\left(\mathcal{A}_{q,\underline{s}}^c \middle| \bigcap_{\ell \in \underline{\mathcal{Q}}_{q,\mathcal{S},\underline{s}}} \mathcal{A}_{\ell,\underline{s}}, \bigcap_{\ell \in \overline{\mathcal{Q}}_{q,\mathcal{S},\underline{s}}} \mathcal{A}_{\ell,\underline{s}}, \bigcap_{k \in \tilde{\mathcal{Q}}_{q,\mathcal{S},\underline{s}}} \mathcal{A}_{k,\underline{s}}^c\right) = \prod_{q \in \mathcal{S}}\left(1 - \Pr\left(\mathcal{A}_{q,\underline{s}} \middle| \bigcap_{\ell \in \underline{\mathcal{Q}}_{q,\mathcal{S},\underline{s}}} \mathcal{A}_{\ell,\underline{s}}, \bigcap_{\ell \in \overline{\mathcal{Q}}_{q,\mathcal{S},\underline{s}}} \mathcal{A}_{\ell,\underline{s}}, \bigcap_{k \in \tilde{\mathcal{Q}}_{q,\mathcal{S},\underline{s}}} \mathcal{A}_{k,\underline{s}}^c\right)\right)
\tag{49}
$$

$$
\begin{aligned}
\Pr\left(\mathcal{E}_{\underline{s},\mathcal{S}}\right) = \prod_{q \in \mathcal{S}^c} g\left(\sum_{\substack{j \in [0:q-1] \\ \underline{s}(q) \neq \underline{s}(j)}} b_j - \sum_{\substack{k \in [0:p-1] \\ \underline{s}(p) \neq \underline{s}(k)}} b_k\right) \\
\times \prod_{q \in \mathcal{S}}\left[1 - \left[\mathbb{1}_{\left\{\overline{\mathcal{Q}}_{q,\mathcal{S},\underline{s}} \neq \emptyset\right\}} + \mathbb{1}_{\left\{\overline{\mathcal{Q}}_{q,\mathcal{S},\underline{s}} = \emptyset\right\}} \mathbb{1}_{\left\{\tilde{\mathcal{Q}}_{q,\mathcal{S},\underline{s}} = \emptyset\right\}} g\left(\sum_{\substack{j \in [0:q-1] \\ \underline{s}(q) \neq \underline{s}(j)}} b_j - \sum_{\substack{k \in [0:p-1] \\ \underline{s}(p) \neq \underline{s}(k)}} b_k\right)\right]\right]
\end{aligned}
\tag{53}
$$

The quantity $\frac{1}{K_a^{c_1-1}-1} \to 0$ as $K_a \to \infty$ since $c_1 > 1$. The quantity $\frac{2^{(n-1)\log_2 K_a}}{2^P} \to 0$ if $P = (n-1)\log_2 K_a + h(K_a)$, where $h(\cdot)$ is any function that grows unbounded with its argument. For instance, if we choose $h(K_a) = \delta \log_2 K_a$ for some fixed constant $\delta > 0$, we get $p_{\text{tree}} \leq \frac{1}{K_a^{c_1-1}-1} + \frac{1}{K_a^{\delta}}$, which implies $p_{\text{tree}} \to 0$ as $K_a \to \infty$.

When the total number of parity-check bits is $P = (n-1+\delta)\log_2 K_a$, we have $M = nJ = B + (n-1+\delta)\log_2 K_a$. Substituting $J = c_1 \log_2 K_a$ into the above equation, we obtain $n = \frac{B+(\delta-1)\log_2 K_a}{(c_1-1)\log_2 K_a}$. Hence, in regime (i), we have $n = c_2 = \frac{\alpha+\delta-1}{c_1-1}$. In regime (ii), this leads to $n = c_2 \frac{K_a}{\log_2 K_a}$, where $c_2 \approx \frac{1}{c_1-1}$ for large $K_a$. These results imply that the number of sub-blocks needed is $\mathcal{O}(1)$ in regime (i); and it scales nearly linearly with $K_a$ in regime (ii).

Let $N_s$ denote the number of channel uses allocated to each CS sub-problem. Standard results in the compressed sensing literature offer sufficient conditions on $N_s$ for exact support recovery using conventional CS algorithms such as LASSO and OMP [17], [32]. For instance, if the measurement matrix is from the uniform Gaussian ensemble, then

$$N_s = \mathcal{O}(K_a \log_2(2^J - K_a)) = \mathcal{O}(K_a J)$$

channel uses are sufficient to ensure that the error probability of the support recovery process is

$$\mathcal{O}(\beta_1 \exp(-\beta_2 \min(2^J, K_a)))$$

when LASSO is employed with a proper choice of the regularization parameter. Parameters $\beta_1$ and $\beta_2$ are fixed constants in the expression above [17]. Altogether, having $N_s = \lambda K_a J$ for some fixed constant $\lambda$ ensures that $np_{\text{cs}} \to 0$ for both regimes (i) and (ii).

Since there are $n$ CS sub-problems, the relation between $N_s$ and the total number of channel uses is $N = nN_s = n\lambda K_a J$. This yields $N = \lambda \frac{(\alpha+\delta-1)c_1}{c_1-1} K_a \log_2 K_a$ for regime (i), and $N \approx \lambda \frac{c_1}{c_1-1} K_a^2$ for regime (ii), both of which are consistent with the statement of the theorem. The computational complexity of the overall scheme is dictated by its two components: the complexity of solving the CS sub-problems, and the expected complexity associated with the tree decoding process.

*a) CS Sub-Problems:* The computational complexity of conventional CS solvers (e.g., LASSO, OMP, NNLS) scales polynomially in the length of the unknown vector. Since we have $n$ sub-problems, the computational complexity of the CS sub-problems is given by

$$C_{\text{cs}} = \mathcal{O}\left(n\left(2^J\right)^\rho\right) = \mathcal{O}\left(nK_a^{\rho c_1}\right)$$

where $\rho \geq 1$. The above expression simplifies to $C_{\text{cs}} = \mathcal{O}\left(K_a^{\rho c_1}\right)$ for regime (i) and $C_{\text{cs}} = \mathcal{O}\left(\frac{K_a^{\rho c_1+1}}{\log_2 K_a}\right)$ for regime (ii). Hence, the computational cost of solving the CS sub-problems is bounded by a polynomial in $K_a$ for both regimes. If CS solvers with linear complexity in the length of the unknown vector are employed for these sub-problems (i.e., $\rho = 1$), then $C_{\text{cs}}$ scales as $K_a^{c_1}$ and $\frac{K_a^{c_1+1}}{\log_2 K_a}$ in regimes (i) and (ii), respectively.

*b) Tree Decoder:* Under regime (i), we can leverage the identity $n = c_2 = \frac{\alpha+\delta-1}{c_1-1}$. With parameters $J = c_1 \log_2 K_a$ and $P = (n-1+\delta)\log_2 K_a$, we can rewrite the boundary of (56) as

$$n - \frac{P}{J} = n - \frac{n-1+\delta}{c_1} = \frac{\alpha}{c_1}.$$

This yields the compact form

$$l_j = \begin{cases} 0 & j < \alpha/c_1 \\ c_1 \log_2 K_a & j \geq \alpha/c_1. \end{cases} \quad (57)$$

The product $\prod_{\ell=q}^{j} p_\ell$ appearing in (10) can then be computed using (57),

$$\prod_{\ell=q}^{j} p_\ell = \begin{cases} 1 & q \leq j < \alpha/c_1 \\ K_a^{-c_1(j+1)+\alpha} & q < \alpha/c_1 \leq j \\ K_a^{-c_1(j+1-q)} & \alpha/c_1 \leq q \leq j. \end{cases} \quad (58)$$

We get an upper bound on the expected complexity for this scenario by substituting (57), (58), $J = c_1 \log_2 K_a$, and $P = (n-1+\delta)\log_2 K_a$ into (10). Recall that $\mathbb{E}[C_{\text{tree}}]$ is of the form

$$K_a \sum_{j=1}^{n-1} l_j + K_a \sum_{j=1}^{n-2} l_{j+1} \sum_{q=1}^{j} K^{j-q}(K-1) \prod_{\ell=q}^{j} p_\ell \quad (59)$$

when $K = K_a$. We recognize the first summand as $K_a P$ and, therefore,

$$K_a \sum_{j=1}^{n-1} l_j = K_a(n-1+\delta)\log_2 K_a = \mathcal{O}\left(K_a \log_2 K_a\right). \quad (60)$$

We break down the second summand along the cases of (58). The first portion of the triplet is

$$K_a \sum_{j=1}^{\alpha/c_1-1} l_{j+1} \sum_{q=1}^{j} K_a^{j-q}(K_a-1) \prod_{\ell=q}^{j} p_\ell$$

$$= K_a c_1 \log_2(K_a) \sum_{q=1}^{\alpha/c_1-1} K_a^{\alpha/c_1-1-q}(K_a-1)$$

$$\leq c_1 K_a^{\alpha/c_1} \log_2 K_a = \mathcal{O}\left(K_a^{\alpha/c_1} \log_2 K_a\right). \quad (61)$$

We note that, among the indices of the first sum in (61), $l_{j+1}$ is non-zero only when $j = \alpha/c_1 - 1$. This greatly simplifies the expression and leads to the succinct characterization. The second portion of the triplet is

$$K_a \sum_{j=\frac{\alpha}{c_1}}^{n-2} l_{j+1} \sum_{q=1}^{\frac{\alpha}{c_1}-1} K_a^{j-q}(K_a-1) \prod_{\ell=q}^{j} p_\ell$$

$$= K_a \sum_{j=\frac{\alpha}{c_1}}^{n-2} c_1 \log_2(K_a) \sum_{q=1}^{\frac{\alpha}{c_1}-1} K_a^{j-q}(K_a-1) K_a^{-c_1(j+1)+\alpha}$$

$$\leq c_1 \frac{K_a^{\alpha/c_1} \log_2 K_a}{K_a^{c_1-1}-1} = \mathcal{O}\left(\frac{K_a^{\alpha/c_1} \log_2 K_a}{K_a^{c_1-1}-1}\right). \quad (62)$$

While tedious, the manipulations in (62) revolve around the evaluation of geometric sums and the application of straightforward upper bounds. This same strategy is applied to the last portion of the sum, with

$$
\begin{aligned}
K_{\mathrm{a}} & \sum_{j=\frac{\alpha}{c_1}}^{n-2} l_{j+1} \sum_{q=\frac{\alpha}{c_1}}^{j} K_{\mathrm{a}}^{j-q}(K_{\mathrm{a}} - 1) \prod_{\ell=q}^{j} p_\ell \\
&= K_{\mathrm{a}}(K_{\mathrm{a}} - 1) \sum_{j=\frac{\alpha}{c_1}}^{n-2} c_1 \log_2(K_{\mathrm{a}}) \sum_{q=\frac{\alpha}{c_1}}^{j} K_{\mathrm{a}}^{j-q} K_{\mathrm{a}}^{-c_1(j+1-q)} \\
&\leq c_1 \left( n - 1 - \frac{\alpha}{c_1} \right) \frac{K_{\mathrm{a}}}{K_{\mathrm{a}}^{c_1 - 1} - 1} \log_2 K_{\mathrm{a}} \\
&= \mathcal{O}\left( \frac{K_{\mathrm{a}} \log_2 K_{\mathrm{a}}}{K_{\mathrm{a}}^{c_1 - 1} - 1} \right).
\end{aligned}
\tag{63}
$$

Collecting the first summand and the three partial sums of the second summand, we gather that

$$
\begin{aligned}
\mathbb{E}[C_{\mathrm{tree}}] &= \mathcal{O}(K_{\mathrm{a}} \log_2 K_{\mathrm{a}}) + \mathcal{O}\left( K_{\mathrm{a}}^{\alpha/c_1} \log_2 K_{\mathrm{a}} \right) \\
&+ \mathcal{O}\left( \frac{K_{\mathrm{a}}^{\alpha/c_1} \log_2 K_{\mathrm{a}}}{K_{\mathrm{a}}^{c_1 - 1} - 1} \right) + \mathcal{O}\left( \frac{K_{\mathrm{a}} \log_2 K_{\mathrm{a}}}{K_{\mathrm{a}}^{c_1 - 1} - 1} \right).
\end{aligned}
\tag{64}
$$

The above expression is dominated by the contribution of its second term. Consequently, the average complexity of tree decoding in regime (i) has an order complexity of $\mathcal{O}\left( K_{\mathrm{a}}^{\alpha/c_1} \log_2 K_{\mathrm{a}} \right)$.

Under regime (ii), we inherit the identity $n = \frac{1}{c_1 - 1} \frac{K_{\mathrm{a}}}{\log_2 K_{\mathrm{a}}} + \frac{\delta - 1}{c_1 - 1}$. Paralleling the steps above, we can rewrite the boundary of (56) as

$$
n - \frac{P}{J} = \frac{(c_1 - 1)n}{c_1} - \frac{\delta - 1}{c_1} = \frac{K_{\mathrm{a}}}{c_1 \log_2 K_{\mathrm{a}}}.
$$

We can avoid duplicating most of the derivations performed under regime (i) by recognizing that the boundary condition above can be obtained from (57) by substituting $\alpha = \frac{K_{\mathrm{a}}}{\log_2 K_{\mathrm{a}}}$. We can discern where the modified formulation has an impact beyond substitution by carefully tracking steps. For the first summand in the expansion of (59), we get

$$
\begin{aligned}
K_{\mathrm{a}} \sum_{j=1}^{n-1} l_j &= K_{\mathrm{a}}(n - 1 + \delta) \log_2 K_{\mathrm{a}} \\
&= K_{\mathrm{a}} \left( c_2 \frac{K_{\mathrm{a}}}{\log_2 K_{\mathrm{a}}} - 1 + \delta \right) \log_2 K_{\mathrm{a}} = \mathcal{O}\left( K_{\mathrm{a}}^2 \right).
\end{aligned}
\tag{65}
$$

The first triplet of the second summand in (61) is unaltered beyond the aforementioned substitution. This yields

$$
\mathcal{O}\left( K_{\mathrm{a}}^{\frac{K_{\mathrm{a}}}{c_1 \log_2 K_{\mathrm{a}}}} \log_2 K_{\mathrm{a}} \right) = \mathcal{O}\left( 2^{K_{\mathrm{a}}/c_1} \log_2 K_{\mathrm{a}} \right),
\tag{66}
$$

where we have leveraged the identity $x^{\frac{x}{\log_2 x}} = 2^x$ for all $x > 0$. The second portion of the triplet is also unaltered beyond substitution. Leveraging the same identity, (62) becomes

$$
\mathcal{O}\left( \frac{K_{\mathrm{a}}^{\frac{K_{\mathrm{a}}}{c_1 \log_2 K_{\mathrm{a}}}} \log_2 K_{\mathrm{a}}}{K_{\mathrm{a}}^{c_1 - 1} - 1} \right) = \mathcal{O}\left( \frac{2^{K_{\mathrm{a}}/c_1} \log_2 K_{\mathrm{a}}}{K_{\mathrm{a}}^{c_1 - 1} - 1} \right)
\tag{67}
$$

under regime (ii). The final component comes from the upper bound in (63). After proper substitution, this component changes to

$$
\begin{aligned}
c_1 & \left( n - 1 - \frac{K_{\mathrm{a}}}{c_1 \log_2 K_{\mathrm{a}}} \right) \frac{K_{\mathrm{a}}}{K_{\mathrm{a}}^{c_1 - 1} - 1} \log_2 K_{\mathrm{a}} \\
&= c_1 \left( c_2 \frac{K_{\mathrm{a}}}{\log_2 K_{\mathrm{a}}} - 1 - \frac{K_{\mathrm{a}}}{c_1 \log_2 K_{\mathrm{a}}} \right) \frac{K_{\mathrm{a}}}{K_{\mathrm{a}}^{c_1 - 1} - 1} \log_2 K_{\mathrm{a}} \\
&= \mathcal{O}\left( \frac{K_{\mathrm{a}}^2}{K_{\mathrm{a}}^{c_1 - 1} - 1} \right).
\end{aligned}
\tag{68}
$$

Aggregating the contributions from these four parts, we get

$$
\begin{aligned}
\mathbb{E}[C_{\mathrm{tree}}] &= \mathcal{O}\left( K_{\mathrm{a}}^2 \right) + \mathcal{O}\left( 2^{K_{\mathrm{a}}/c_1} \log_2 K_{\mathrm{a}} \right) \\
&+ \mathcal{O}\left( \frac{2^{K_{\mathrm{a}}/c_1} \log_2 K_{\mathrm{a}}}{K_{\mathrm{a}}^{c_1 - 1} - 1} \right) + \mathcal{O}\left( \frac{K_{\mathrm{a}}^2}{K_{\mathrm{a}}^{c_1 - 1} - 1} \right).
\end{aligned}
\tag{69}
$$

The above expression is dominated by the second term. Thus, the average complexity of tree decoding under regime (ii) is on the order of $\mathcal{O}\left( 2^{K_{\mathrm{a}}/c_1} \log_2 K_{\mathrm{a}} \right)$. This completes the proof of Theorem 8.

### B. Proof of Theorem 9

In this section, we provide a proof for Theorem 9. We wish to study the scenario where parity bits are assigned uniformly across sub-blocks under the simplifying conditions of Assumption 3. For the uniform allocation, these conditions reduce to $\underline{w}_i(j) \neq \underline{w}_k(j)$ for all $j \in [0 : n - 1]$. To make this precise, we find circumstances under which the probability of Assumption 3 being violated goes to zero as $K_{\mathrm{a}}$ grows unbounded. We begin by establishing an upper bound on the expected number of sub-blocks for which the uniqueness condition is violated. The expected number of fragments that are repeated more than once can the be bounded by

$$
\begin{aligned}
\mathbb{E}[R] &= \mathbb{E}\left[ \sum_{i=1}^{K_{\mathrm{a}}} \sum_{k=1}^{i-1} \sum_{j=0}^{n-1} \mathbb{1}_{\left\{ \underline{w}_i(j) = \underline{w}_k(j) \right\}} \right] \\
&= \sum_{i=1}^{K_{\mathrm{a}}} \sum_{k=1}^{i-1} \sum_{j=0}^{n-1} \Pr\left( \underline{w}_i(j) = \underline{w}_k(j) \right) \leq \frac{n K_{\mathrm{a}}^2}{2^b}.
\end{aligned}
$$

Then, by the Markov inequality, we get

$$
\Pr(R \geq 1) \leq \mathbb{E}[R] \leq \frac{n K_{\mathrm{a}}^2}{2^b}.
\tag{70}
$$

It remains to choose suitable parameters for $\alpha$, $c_1$, and $c_2$ to ensure that (70) vanishes as $K_{\mathrm{a}} \to \infty$. The selection of suitable parameters depends on the asymptotic regime under consideration. Thus, we treat the cases associated with regime (i) and regime (ii) separately.

*a) Regime (i):* In this case, the number of information bits per non-root sub-block is dictated by the first column in Table II, with

$$
b = \frac{B - J}{n - 1} = \frac{\alpha - c_1}{c_2 - 1} \log_2 K_{\mathrm{a}}.
\tag{71}
$$

The total number of parity bits for this setting becomes $P = nJ - B = (c_1\, c_2 - \alpha) \log_2 K_a$ and, consequently, the number of parity bits per non-root sub-block becomes

$$l = \frac{nJ - B}{n - 1} = \frac{c_1 c_2 - \alpha}{c_2 - 1} \log_2 K_a. \tag{72}$$

Substituting the expression for $b$ in (70), we get

$$\Pr(R \geq 1) \leq \frac{c_2 K_a^2}{2^{\frac{\alpha - c_1}{c_2 - 1} \log_2 K_a}} = c_2 K_a^{2 - \frac{\alpha - c_1}{c_2 - 1}}. \tag{73}$$

Picking $c_1$ and $c_2$ such that $\frac{\alpha - c_1}{c_2 - 1} > 2$ meets our requirements. In words, limiting the number of sub-blocks $n = c_2$ simultaneously ensures that the number of information bits per fragment $b$ is large and, hence, the probability of a match $2^{-b}$ is low. At the same time, the number of sub-blocks must be large enough to account for all the information bits while also leaving room for parity bits, $c_1\, c_2 - \alpha > 0$.

*b) Regime (ii):* The number of information bits for this second case is characterized by

$$
\begin{aligned}
b &= \frac{B - J}{n - 1} = \frac{K_a - c_1 \log_2 K_a}{c_2 \frac{K_a}{\log_2 K_a} - 1} \\
&\geq \frac{1}{c_2} \log_2 K_a - \frac{c_1}{c_2} \frac{(\log_2 K_a)^2}{K_a} \\
&= \frac{1}{c_2} \log_2 K_a - o(1). 
\end{aligned} \tag{74}
$$

The number of parity bits in this setting becomes $P = nJ - B = (c_1\, c_2 - 1)K_a$, which yields

$$
\begin{aligned}
l &= \frac{nJ - B}{n - 1} = \frac{c_1 c_2 - 1}{c_2 \frac{K_a}{\log_2 K_a} - 1} K_a \\
&\geq \frac{c_1 c_2 - 1}{c_2} \log_2 K_a.
\end{aligned} \tag{75}
$$

After incorporating this admittedly intricate inequality governing $b$ in (70), we obtain

$$\Pr(R \geq 1) \leq \frac{c_2 \frac{K_a}{\log_2 K_a} K_a^2}{2^{\frac{1}{c_2} \log_2 K_a - o(1)}} = c_2 \frac{K_a^{3 - \frac{1}{c_2}}}{\log_2 K_a} 2^{o(1)}. \tag{76}$$

Choosing $c_1$ and $c_2$ such that $c_2 < 1/3$ will make the right-hand side of this expression go to zero as $K_a \to \infty$. At the same time, the number of sub-blocks cannot grow too fast. Furthermore, the number of sub-blocks must be large enough to account for all the information bits and, simultaneously, leave room for parity bits. This is accomplished with $c_1\, c_2 > 1$.

*1) Asymptotic Reliability:* The conditions outlined above control the number of possible matches, and it pushes $\Pr(R \geq 1)$ to zero. In other words, under the stipulations included in the statement of Theorem 9, we know that the conditions of Assumption 3 are met with probability approaching one as $K_a \to \infty$. Through conditioning, we can therefore focus on cases where Assumption 3 holds. A (vanishingly small) decoding error can be declared when the uniqueness conditions are not met by the output of the CS algorithms. This strategy allows us to characterize computational complexity while operating under Assumption 3, at the expense of an additional $\Pr(R \geq 1)$ term contributing to the error probability. This idiosyncratic situation has no impact on asymptotic

performance when system parameters are selected judiciously. Pragmatically, this enables us to use the simplified expressions of Section IV-B in computing true asymptotic behavior.

For ease of exposition, we take the total number of parity bits $P$ to be an integer multiple of $n - 1$. With careful accounting, it is possible to relax this assumption and retain the character of our findings. Still, this structure simplifies the proof considerably. For the current scenarios, the number of parity bits per non-root sub-block is given by

$$l_j = \frac{P}{n - 1} \qquad j \in [1 : n - 1]. \tag{77}$$

The quantity $\prod_{\ell = q}^{n - 1} p_\ell$ appearing in (9) can then be computed using (77),

$$\prod_{\ell = q}^{n - 1} p_\ell = \left( \frac{1}{2^{\frac{P}{n - 1}}} \right)^{n - q} = 2^{-\frac{P(n - q)}{n - 1}}.$$

Substituting this expression into (9), we get an upper bound for the expected value $\mathbb{E}[L_{n - 1}]$,

$$
\begin{aligned}
\mathbb{E}[L_{n - 1}] &= \sum_{q = 1}^{n - 1} K_a^{n - 1 - q}(K_a - 1) \frac{1}{2^{\frac{P(n - q)}{n - 1}}} \\
&\leq \frac{K_a}{2^l - K_a}.
\end{aligned} \tag{78}
$$

For asymptotic reliability of the CCS algorithm, we need the expected number of erroneous paths that survive at the end of the decoding process to go to zero. Again, we find suitable conditions for the two regimes separately.

*a) Regime (i):* In this regime, $l$ is given by (72). Substituting this expression in (78), we get the upper bound

$$\mathbb{E}[L_{n - 1}] \leq \frac{1}{K_a^{\frac{c_1\, c_2 - \alpha}{c_2 - 1} - 1} - 1}.$$

This quantity and $p_{\text{tree}}$ go to zero as $K_a \to \infty$ provided that $\frac{\alpha - c_1}{c_2 - 1} < c_1 - 1$. Intuitively, the discriminating power of parity bits has to counteract the per stage growth of the tree.

*b) Regime (ii):* In this case, $l$ is governed by the inequality in (75) and, after substitution and manipulations, we obtain

$$\mathbb{E}[L_{n - 1}] \leq \frac{K_a}{K_a^{\frac{c_1\, c_2 - 1}{c_2}} - K_a}.$$

The expected number of extraneous paths and $p_{\text{tree}}$ tend to zero as $K_a \to \infty$ whenever $c_1 - 1 > 1/c_2$. This condition again arises from the tension between the growth at every stage in the tree and the ability of parity bits to identify erroneous candidate paths.

*2) Complexity Analysis:* In a manner akin to the proof of Theorem 8, the number of channel uses for the CS sub-problem is chosen as $N_s = \lambda K_a J$. This ensures that the probability of failure for one CS sub-component goes to zero exponentially fast and, hence, $np_{cs} \to 0$ for both regimes identified above. The total number of channel uses $N$ is then given by $N = nN_s = c_1\, n\lambda K_a \log_2 K_a$. This expression simplifies to $N = c_1\, c_2 \lambda K_a \log_2 K_a$ under regime (i); and it becomes $N = c_1\, c_2 \lambda K_a^2$ under regime (ii). We note that both cases are consistent with the statement of the theorem. As

before, the computational complexity for the CCS algorithm arises from two components: the complexity of solving the CS sub-problems and the average complexity associated with the tree decoder.

*a) CS Sub-Problems:* The analysis of this component is similar to that contained in the proof of Theorem 8. The CS sub-problems have polynomial complexity in the length of the unknown vector. The complexity of the CS sub-problems $C_{cs}$ can then be expressed as

$$C_{cs} = \mathcal{O}\left(n\left(2^J\right)^\rho\right) = \mathcal{O}\left(nK_a^{\rho c_1}\right)$$

where $\rho \geq 1$. Therefore, $C_{cs}$ scales as $\mathcal{O}\left(K_a^{c_1}\right)$ for regime (i) and $\mathcal{O}\left(\frac{K_a^{c_1+1}}{\log_2 K_a}\right)$ for regime (ii) when a CS algorithm of linear complexity is employed to solve the CS sub-problems.

*b) Tree Decoder:* For uniformly distributed parity bits, the expected computational complexity of (10) can be simplified to

$$\mathbb{E}[C_{tree}] = K_a \sum_{j=1}^{n-1} l + K_a \sum_{j=1}^{n-2} l \sum_{q=1}^{j} K_a^{j-q}(K_a - 1) \prod_{\ell=q}^{j} 2^{-l}$$

$$\leq K_a P + \frac{K_a(K_a - 1)}{2^l - K_a} P.$$

From our characterization of $\mathbb{E}[L_{n-1}]$ and its upper bound in (78), we know that

$$\frac{K_a}{2^l - K_a} = o(1)$$

whenever parameters are selected judiciously. In regime (i), $P = \mathcal{O}(\log_2 K_a)$ and, hence, $\mathbb{E}[C_{tree}] = \mathcal{O}(K_a \log_2 K_a)$. Likewise, in regime (ii), $P = \mathcal{O}(K_a)$ and $\mathbb{E}[C_{tree}] = \mathcal{O}\left(K_a^2\right)$. These results are those reported in the statement of Theorem 9.

## References

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

[2] E. Paolini, C. Stefanovic, G. Liva, and P. Popovski, "Coded random access: Applying codes on graphs to design random access protocols," *IEEE Commun. Mag.*, vol. 53, no. 6, pp. 144–150, Jun. 2015.

[3] X. Chen, T.-Y. Chen, and D. Guo, "Capacity of Gaussian many-access channels," *IEEE Trans. Inf. Theory*, vol. 63, no. 6, pp. 3516–3539, Jun. 2017.

[4] O. Ordentlich and Y. Polyanskiy, "Low complexity schemes for the random access Gaussian channel," in *Proc. IEEE Int. Symp. Inf. Theory*, Jun. 2017, pp. 2528–2532.

[5] A. Vem, K. R. Narayanan, J. Cheng, and J.-F. Chamberland, "A user-independent serial interference cancellation based coding scheme for the unsourced random access Gaussian channel," in *Proc. IEEE Inf. Theory Workshop*, Nov. 2017, pp. 121–125.

[6] Y. Jin, Y.-H. Kim, and B. D. Rao, "Limits on support recovery of sparse signals via multiple-access communication techniques," *IEEE Trans. Inf. Theory*, vol. 57, no. 12, pp. 7877–7892, Dec. 2011.

[7] J.-P. Hong, W. Choi, and B. D. Rao, "Sparsity controlled random multiple access with compressed sensing," *IEEE Trans. Wireless Commun.*, vol. 14, no. 2, pp. 998–1010, Feb. 2015.

[8] L. Zhang, J. Luo, and D. Guo, "Neighbor discovery for wireless networks via compressed sensing," *Perform. Eval.*, vol. 70, nos. 7–8, pp. 457–471, Jul. 2013.

[9] G. Wunder, H. Boche, T. Strohmer, and P. Jung, "Sparse signal processing concepts for efficient 5G system design," *IEEE Access*, vol. 3, pp. 195–208, 2015.

[10] X. Chen, D. Guo, and G. W. Wornell, "Sparse OFDM: A compressive sensing approach to asynchronous neighbor discovery," 2017, *arXiv:1706.09387*. [Online]. Available: http://arxiv.org/abs/1706.09387

[11] S. Pawar and K. Ramchandran, "Computing a $k$-sparse $n$-length discrete Fourier transform using at most $4k$ samples and $O(k \log k)$ complexity," in *Proc. IEEE Int. Symp. Inf. Theory*. Jul. 2013, pp. 464–468.

[12] V. K. Amalladinne, K. R. Narayanan, J.-F. Chamberland, and D. Guo, "Asynchronous neighbor discovery using coupled compressive sensing," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, May 2019, pp. 4569–4573.

[13] G. Wunder, C. Stefanovic, P. Popovski, and L. Thiele, "Compressive coded random access for massive MTC traffic in 5G systems," in *Proc. 49th Asilomar Conf. Signals, Syst. Comput.*, Nov. 2015, pp. 13–17.

[14] F. Fazel, M. Fazel, and M. Stojanovic, "Random access compressed sensing over fading and noisy communication channels," in *Proc. IEEE Int. Symp. Inf. Theory Proc.*, Jul. 2012, pp. 2114–2125.

[15] A. C. Gilbert, Y. Li, E. Porat, and M. J. Strauss, "For-all sparse recovery in near-optimal time," *ACM Trans. Algorithms*, vol. 13, no. 3, pp. 1–26, Aug. 2017.

[16] K. G. Larsen, J. Nelson, H. L. Nguyen, and M. Thorup, "Heavy hitters via cluster-preserving clustering," in *Proc. IEEE Symp. Found. Comput. Sci.* Oct. 2016, pp. 61–70.

[17] M. J. Wainwright, "Sharp thresholds for high-dimensional and noisy sparsity recovery using $\ell_1$-constrained quadratic programming (Lasso)," *IEEE Trans. Inf. Theory*, vol. 55, no. 5, pp. 2183–2202, May 2009.

[18] M. J. Wainwright, "Information-theoretic limits on sparsity recovery in the high-dimensional and noisy setting," *IEEE Trans. Inf. Theory*, vol. 55, no. 12, pp. 5728–5741, Dec. 2009.

[19] W. Wang, M. J. Wainwright, and K. Ramchandran, "Information-theoretic limits on sparse signal recovery: Dense versus sparse measurement matrices," *IEEE Trans. Inf. Theory*, vol. 56, no. 6, pp. 2967–2979, Jun. 2010.

[20] R. Tibshirani, "Regression shrinkage and selection via the lasso," *J. Roy. Stat. Soc. B, Methodol.*, vol. 58, no. 1, pp. 267–288, Jan. 1996.

[21] T. Blumensath and M. E. Davies, "Iterative hard thresholding for compressed sensing," *Appl. Comput. Harmon. Anal.*, vol. 27, no. 3, pp. 265–274, Nov. 2009.

[22] A. C. Gilbert, M. J. Strauss, J. A. Tropp, and R. Vershynin, "One sketch for all: Fast algorithms for compressed sensing," in *Proc. Symp. Theory Comput.*, 2007, pp. 237–246.

[23] A. Gilbert and P. Indyk, "Sparse recovery using sparse matrices," *Proc. IEEE*, vol. 98, no. 6, pp. 937–947, Jun. 2010.

[24] P. Indyk, "Explicit constructions for compressed sensing of sparse signals," in *Proc. ACM-SIAM Symp. Discrete Algorithms*, 2008, pp. 30–33.

[25] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge, U.K.: Cambridge Univ. Press, 2004.

[26] C. L. Lawson and R. J. Hanson, *Solving Least Squares Problems*, vol. 15. Philadelphia, PA, USA: SIAM, 1995.

[27] M. Grant, S. Boyd, and Y. Ye. (2008). *CVX: MATLAB Software for Disciplined Convex Programming*. [Online]. Available: http://cvxr.com/cvx

[28] V. K. Amalladinne, A. Vem, D. K. Soma, K. R. Narayanan, and J.-F. Chamberland, "A coupled compressive sensing scheme for unsourced multiple access," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, Apr. 2018, pp. 6628–6632.

[29] A. Fengler, P. Jung, and G. Caire, "SPARCs and AMP for unsourced random access," in *Proc. IEEE Int. Symp. Inf. Theory*, Jul. 2019, pp. 2843–2847.

[30] R. Calderbank and A. Thompson, "CHIRRUP: A practical algorithm for unsourced multiple access," *Inf. Inference, J. IMA*, no. iaz029, 2019, doi: 10.1093/imaiai/iaz029.

[31] R. L. Graham, D. E. Knuth, and O. Patashnik, *Concrete Mathematics: A Foundation for Computer Science*, 2nd ed. Reading, MA, USA: Addison-Wesley, 1994.

[32] T. T. Cai and L. Wang, "Orthogonal matching pursuit for sparse signal recovery with noise," *IEEE Trans. Inf. Theory*, vol. 57, no. 7, pp. 4680–4688, Jul. 2011.

**Vamsi K. Amalladinne** (Graduate Student Member, IEEE) received the B.Tech. degree in electronics and communication engineering from The LNM Institute of Information Technology, Jaipur, India, in 2012, and the M.Tech. degree in signal processing and communications from IIT Kanpur, India, in 2014. He is currently pursuing the Ph.D. degree with the Department of Electrical and Computer Engineering, Texas A&M University at College Station, College Station, TX, USA. From June 2014 to August 2016, he was employed as a DSP Firmware Developer for CDMA systems at Qualcomm India Pvt., Ltd. His research interests include wireless communications, signal processing, error control coding, and compressed sensing.

**Jean-Francois Chamberland** (Senior Member, IEEE) 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, Texas A&M University. His research interests include computing, information, and inference. He was 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 has served as an Associate Editor for the IEEE TRANSACTIONS ON INFORMATION THEORY from 2017 to 2020.

**Krishna R. Narayanan** (Fellow, IEEE) received the B.E. degree from the Coimbatore Institute of Technology, Coimbatore, India, in 1992, the M.S. degree from Iowa State University, Ames, IA, USA, in 1994, and the Ph.D. degree from the Georgia Institute of Technology, Atlanta, GA, USA, in 1998. He is currently the Eric D. Rubin'06 Professor with the Electrical and Computer Engineering, Texas A&M University at College Station, College Station, TX, USA. His research interests include coding theory, information theory, and signal processing with applications to wireless communications, data storage, and data science. He has served as an Associate Editor for the IEEE TRANSACTIONS ON INFORMATION THEORY from 2015 to 2018 and 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.