A COUPLED COMPRESSIVE SENSING SCHEME FOR UNSOURCED MULTIPLE ACCESS

Vamsi K. Amalladinne, Avinash Vem, Dileep Kumar Soma,
Krishna R. Narayanan, Jean-Francois Chamberland

Department of Electrical and Computer Engineering, Texas A&M University

ABSTRACT

We present a novel divide-and-conquer compressive sensing (CS) based approach for the unsourced random access problem [1]. Each user's data is split into several sub-blocks and a systematic linear block code is used to introduce redundancy into this data. CS based encoder is then employed at each sub-block before transmission and at the decoder, the outputs are combined using a low-complexity tree based algorithm. We demonstrate that the proposed scheme outperforms all the existing practical coding schemes in literature and is only ≈ 4.3dB away from the Polyanskiy's achievability limit [1].

***Index Terms*—** unsourced multiple-access, uncoordinated multiple-access, compressive sensing, tree encoding

1. INTRODUCTION

The unsourced multiple access problem, initially proposed by Polyanskiy [1], is a novel and interesting twist to the uncoordinated multiple access problem [2, 3]. In this paradigm, there are a total of K_{tot} users in the system out of which, at any given time, a maximum of K_{a} users want to transmit a B-bit message to the access point. The access point is interested in recovering only the set of messages being transmitted without regard to the identity of the respective sources. With regard to the system parameters, the total number of users K_{tot} can be very large whereas K_{a} and B are fairly small, typically a few hundreds.

Since the regime of interest for the number of message bits and consequently the block length, is extremely small, non-asymptotic information-theoretic benchmarks are required. Polyanskiy [1] derived finite block length achievability bounds for the unsourced MAC based on random Gaussian code book and the optimal minimum mean-squared error decoder. In [4], Ordentlich and Polyanskiy observed that several existing multiple access strategies perform poorly, especially, for values of K_{a} larger than 100. They also proposed the first low complexity coding scheme. In their scheme, the transmission period is divided into sub-blocks (or slots) and it is assumed that all the users are aware of the slot synchronization structure. The proposed coding scheme consists of an encoder that transmits a codeword in a randomly chosen slot. This codeword transmitted in a slot is from a concatenated code that is designed for a T-user real addition Gaussian multiple access channel (T-GMAC), typical values of T being 2 to 5. The proposed scheme although performs significantly better than the existing multiple access strategies, there is a significant gap of $\approx$ 20 dB from the achievability limit [1]. In [5], we proposed a low complexity coding scheme relying on the similar slotted structure that: (i) consists of an improved, close-to-optimal coding scheme for the T-GMAC and (ii) leverages slots with more than T users transmitting instead of discarding them and applies successive interference cancellation decoder across slots. The combination of the above features results in significantly improved performance compared to [4] and is only $\approx$ 6 dB away from the above mentioned achievability limit.

Both of the above schemes take a channel coding view of the problem wherein the K_{a}-user GMAC is reduced to multiple smaller T-GMAC channel problems. In this paper, we take a compressed sensing view of the problem. A naive compressed sensing solution to the overall unsourced multiple access problem requires solving for a 2^B-length, K_{a}-sparse vector. This implies sensing matrices to the order of 2^{100} columns, which makes the problem intractable.

The key idea in this paper is to divide the information blocks of the users into smaller sub-blocks such that each sub-block is amenable to a compressed sensing (CS) framework. Before encoding using the CS, each information sub-block is encoded using a systematic linear block code. Given the output from CS sparse recovery algorithm in each sub-block, the original message is then recovered by piecing together the outputs, leveraging the redundancy due to the block code in each sub-block, via a low-complexity tree-based algorithm. As far as we are aware the proposed scheme is the best known practical coding scheme to date.

We use the following notation throughout the paper. $\mathbb{R}_+, \mathbb{Z}_+$ and $\mathbb{N}$ to denote the set of non-negative reals, set of non-negative integers and the set of natural numbers respectively. For any $a, b \in \mathbb{Z}_+$ with $a \leq b$, we use $[a : b]$ to denote the set $\{c \in \mathbb{Z}_+ : a \leq c \leq b\}$. We write $X \sim B(n, p)$ if a random variable X follows binomial distribution with parameters $n \in \mathbb{N}$ and $p \in [0, 1]$. For any set A, $|A|$ denotes the cardinality of A. For any $x \in \mathbb{R}$, $[x]$ denotes the nearest integer of x.

2. SYSTEM MODEL

Let $\mathbf{S}_{\rm tot}$ represent the set of devices within the network and $\mathbf{S}_{\rm a}$ denote the subset of active devices seeking to transmit data at a particular time instant, $\mathbf{S}_{\rm a} \subset \mathbf{S}_{\rm tot}$. Let $|\mathbf{S}_{\rm tot}| = K_{\rm tot}$ and $|\mathbf{S}_{\rm a}| = K_{\rm a}$. Each active device wishes to communicate B bits of information to a base station through uncoordinated uplink transmission. Let the number of channel uses dedicated to this process be N. Let $W = \{\vec{w}_k : k \in \mathbf{S}_{\rm a}\}$ be the collection of B bit message vectors associated with the active devices. We assume that devices pick their message vectors 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

$$\vec{\mathrm{y}} = \textstyle\sum_{k\in\mathbf{S}_{\rm a}} \vec{\mathrm{x}}_k + \vec{\mathrm{z}},$$

where $\vec{\mathrm{x}}_k$ is the N-dimensional vector transmitted by device k, and $\vec{z}$ represents additive white Gaussian noise. The signal sent by every device is power constrained, i.e., $\|\vec{\mathrm{x}}_k\|_2^2 \leq NP$ for $k \in \mathbf{S}_{\rm a}$, a scenario akin to [1]. The energy-per-bit is then given by $\frac{E_b}{N_0} \triangleq \frac{NP}{2B}$. The receiver produces an estimate $\widehat{W}(\vec{\mathrm{y}})$ for the list of transmitted binary vectors W with $|\widehat{W}(\vec{\mathrm{y}})| \leq K_{\rm a}$. The per-user error probability of the system is defined as

$$P_{\rm e} = \tfrac{1}{K_{\rm a}} \textstyle\sum_{k\in\mathbf{S}_{\rm a}} \Pr\left(\vec{w}_k \notin \widehat{W}(\vec{\mathrm{y}})\right). \tag{1}$$

We propose an encoding and decoding scheme that achieves $P_{\rm e} \leq \varepsilon$, where ε is the target error probability with manageable computational complexity.

3. PROPOSED SCHEME

The key idea to limit complexity consists in dividing the data stream generated by active devices into several sub-blocks. These sub-blocks, with their very short packet fragments, are then amenable to computationally efficient compressed sensing (CS) algorithms. This process yields a list of likely fragments for every sub-block. The original messages are then recovered by piecing together compatible fragments. Technically, this latter step is accomplished by preemptively adding redundancy to the codewords $\{\vec{x}_k\}$, and then leveraging this redundancy while combining sub-blocks via a low-complexity tree-based algorithm.

The specifics of the coding scheme are detailed below. A notional diagram of the proposed system appears in Fig. 1.

3.1. Encoder

The transmission strategy features two components: a systematic linear block code based on random parity checks, which we refer to as the tree encoder, and a CS encoder. The tree encoder adds the redundancy required to identify and combine the sub-blocks corresponding to a parent message from a pool of candidate fragments; whereas the CS encoder transforms the sub-blocks output by the tree encoder into signals suitable for noisy compressive sensing.

3.1.1. Tree Encoder Based on Random Parity Checks

Every B-bit binary message vector $\vec{w}$ is encoded into M bits using a systematic linear block code, which has random parity check constraints. Algorithmically, a message vector is partitioned into n sub-blocks, with the $i^{\rm th}$ sub-block consisting of m_i message bits, $\sum_{i=0}^{n-1} m_i = B$. The tree encoder appends l_i parity bits to sub-block i, except for the first block as we choose $l_0 = 0$. All the coded sub-blocks have the same length i.e., $m_i + l_i = J \triangleq \frac{M}{n}$, $\forall\ i \in [0 : n-1]$. The parity check bits in each sub-block are constructed as follows. Let $(p_0^{(i)}, p_1^{(i)}, \ldots, p_{l_i-1}^{(i)})$ denote the parity bits in sub-block i. These bits are generated satisfying parity check constraints for all the message bits appearing until the respective sub-block. Towards this end, we concatenate the message bits of all the sub-blocks $k \in [0 : i]$ and index them with the set $[0 : \sum_{k=0}^{i} m_k - 1]$. We then choose l_i subsets $\mathcal{A}_j^{(i)} \subseteq [0 : \sum_{k=0}^{i} m_k - 1]\ \forall\ j \in [0 : l_i - 1]$ uniformly at random without replacement. Now, $p_j^{(i)}$ is chosen as the modulo-2 sum of all the message bits indexed by the set $\mathcal{A}_j^{(i)}$. In effect, $p_j^{(i)}$ acts as a parity check constraint for some randomly chosen message bits appearing till the sub-block i. In Sec. 4.1, we describe an optimization framework for the choice of parity length vector $\vec{l} = (l_0 = 0, l_1 \ldots, l_{n-1})$.

3.1.2. CS Encoder

Let $\mathbf{A} = [\vec{\mathrm{a}}_1, \ldots, \vec{\mathrm{a}}_{2^J}] \in \{\pm\sqrt{P}\}^{\tilde{N}\times 2^J}$, where $\tilde{N} \triangleq \frac{N}{n}$, denote a compressed sensing matrix that is designed such that it can recover any $K_{\rm a}$-sparse binary vector in the presence of noise with a low probability of error. The J bits in a sub-block are encoded using a bijective function $f : \{0,1\}^J \to \{\vec{\mathrm{a}}_j, j \in [1 : 2^J]\}$, which maps each sub-block to a column in $\mathbf{A}$. In other words, if $\vec{\tilde{w}}_k = [\vec{\tilde{w}}_0^{(k)} \vec{\tilde{w}}_1^{(k)} \ldots \vec{\tilde{w}}_{n-1}^{(k)}]$ denotes the binary sequence corresponding to the user k output by the tree encoder, where $\vec{\tilde{w}}_i^{(k)}$ denotes the $i^{\rm th}$ tree encoded sub-block. Then, for each sub-block $\vec{\tilde{w}}_i^{(k)}$, the user transmits a column $\vec{\mathrm{a}}_j$ from the sensing matrix $\mathbf{A}$.

3.2. Decoder

The decoding scheme consists of two components- CS decoder operating in each sub-block and a tree decoder operating across sub-blocks. The CS decoder implements a compressed sensing algorithm to identify the list of sub-blocks transmitted by the active users. The tree decoder tries to group all the sub-blocks that correspond to a particular user together,

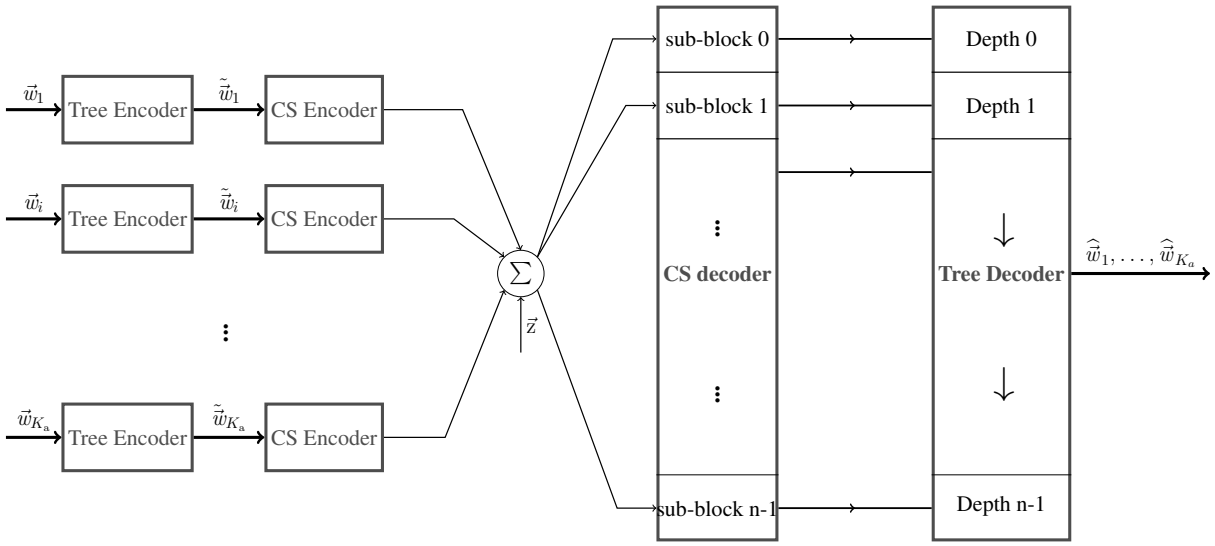

Fig. 1. Schematic of the proposed scheme.

by harnessing the redundancy employed during the encoding process. We describe the two components of the proposed decoder below.

3.2.1. CS Decoder

The signal received during the i^{th} sub-block can be expressed as $\vec{\mathrm{y}}_i = \mathbf{A}\vec{\mathrm{b}}_i + \vec{\mathrm{z}}_i$, where $\vec{\mathrm{b}}_i \in \{0,1\}^{2^J}$ denotes a K_{a}-sparse binary vector that indicates the list of i^{th} sub-blocks transmitted by all the active users. The objective of the CS decoder is to provide an estimate of the sparse vector $\vec{\mathrm{b}}_i$ from the received signal $\vec{\mathrm{y}}_i$. This problem is similar to the support recovery problem in standard compressed sensing literature because of the fact that the non-zero entries of $\vec{b}_i$ are all 1's. We first employ the non-negative least squares (NNLS) algorithm to get an estimate $\vec{b}_i^{(\text{nnls})}$ of the vector $\vec{b}_i$. However, this does not ensure that the entries of the vector $\vec{b}_i^{(\text{nnls})}$ are binary. The final binary estimate $\hat{\vec{b}}_i$ of $\vec{b}_i$ is obtained by setting the K largest entries of the vector $\vec{\mathrm{b}}_i^{(\text{nnls})}$ to 1 and the remaining $2^J - K$ entries to 0. The number K is chosen as $K = K_{\mathrm{a}} + K_\delta$, where K_δ is a small positive integer. Even though the list output by the CS decoder is of size larger than K_{a}, the quantity K_δ is carefully chosen such that the erroneously decoded sub-blocks are unlikely to satisfy all the parity check constraints imposed due to the encoder structure.

3.2.2. Tree Decoder

The tree decoder attempts to find a valid transmitted string corresponding to each user among the various lists output by the CS decoder. Towards this end, the decoder constructs a decoding tree for *each user* as follows. We fix any sub-block from the list of all possible first sub-blocks supplied by the CS decoder as the root node for the tree. When the first sub-block is fixed, there are K possible choices for the second sub-block and these are the nodes which appear in the first stage of the tree. We now have K possible choices for the third sub-block for each choice of second sub-block and hence K^2 nodes in the second stage. This process is continued till the $(n-1)^{\text{th}}$ stage which has K^{n-1} nodes that correspond to the leaf nodes of this tree. For each leaf node, the path connecting itself and the root node is a possible choice for the transmitted string totaling K^{n-1} choices. If there exists a single valid path, the decoder outputs the corresponding message vector else a failure.

The number of paths increases exponentially with the stages of the tree and hence a naive search through all the leaf nodes is infeasible. Hence, we attempt to prune the tree at each stage by retaining fewer number of paths. At stage $i \geq 1$, the decoder retains only those nodes that satisfy the l_i bit parity constraints on all the message bits appearing till that stage. This process is continued till the $(n-1)^{\text{th}}$ stage. The Complexity of this decoding scheme depends on the number of nodes surviving each stage, since parity checks have to be enforced only on the children of surviving nodes in the next stage of the tree decoding process. A comprehensive analysis of probability of decoding failure and the decoding complexity is given in the following section.

Remark 1 (Iterative extension)**.** *The successful outputs from the tree decoder can be subtracted off from the respective received signals in each sub-block. This can potentially improve the estimate provided by the CS decoder compared to the previous iteration and this process can be repeated iteratively until the gains become insignificant.*

4. PERFORMANCE ANALYSIS

Let us assume that the list output by the CS decoder contains the sub-block i transmitted by user k with a probability $1-p_{cs}$, and with a probability p_{cs}, this block is erroneously replaced by a vector chosen uniformly at random from the set $\{0,1\}^J$. Let E_k denote the event that user k's transmitted binary message is not present in the list output by the tree decoder and C_k denote the event that all the sub-blocks corresponding to this user are present in the lists output by the CS decoder. Then, the quantity $P(E_k)$ can be computed as,

$$P(E_k) = P(E_k|C_k)P(C_k) + P(E_k|\overline{C_k})P(\overline{C_k}). \quad (2)$$

If the CS decoder fails to decode atleast one of the sub-blocks that correspond to a user, then the output of the tree decoder would not contain the original message transmitted by that user and so we have, $P(E_k|\overline{C_k}) = 1$. The quantity $P(C_k)$ can be computed as $P(C_k) = (1-p_{cs})^n$. $E_k|C_k$ is the event that the tree decoder declares a failure because of more than one path surviving the tree decoding process. Let us denote this probability by p_{tree}. When there are no iterations involved in the decoding process, the quantity P_e is the same as $P(E_k)$ and they can now be computed using (2) and the above observations as $P_e = 1-(1-p_{\text{tree}})(1-p_{cs})^n$. We now provide a closed form expression for computing the quantity p_{tree} and also analyze the decoding complexity of this system. Further, we provide an optimization framework for the choice of parity lengths.

Let L_i denote the random variable for the number of erroneous paths that survive stage $i \in [1:n-1]$ of the tree decoding process.

Lemma 2. *Expected value of the quantity L_i is given by,*

$$\mathbf{E}[L_i] = \sum_{m=1}^{i} \left[K^{i-m}(K-1) \prod_{j=m}^{i} p_j \right], i \in [1:n-1], \quad (3)$$

where $p_i = \dfrac{1}{2^{l_i}}, q_i = 1-p_i \; \forall \; i \in [1:n-1]$.

Proof. Let $\vec{v}$ denote a vector chosen uniformly at random from the set of binary sequences $\{0,1\}^m$ for some $m \in \mathbb{N}$. Probability that $\vec{v}$ satisfies l random linearly independent parity checks is given by $p = \frac{1}{2^l}$. At every stage $i \in [1:n-1]$ of the decoding tree, L_i is the number of paths that survive this stage minus 1 (The true path survives all the checks deterministically). At stage 1, probability that any binary random vector of length $2J$ satisfies l_1 randomly chosen parity checks is given by $p_1 = \frac{1}{2^{l_1}}$. Hence, probability that k out of the $K-1$ incorrect paths survive during stage 1 is given by,

$$P(L_1 = k) = \binom{K-1}{k} p_1^k q_1^{K-1-k}.$$

Hence, $L_1 \sim B(K-1, p_1)$. Also, for all $i \geq 2$ given L_{i-1}, L_i is the sum of $L_{i-1}+1$ independent binomial random variables, L_{i-1} of them with parameters (K, p_i) and one of them with parameters $(K-1, p_i)$. Hence $L_i|L_{i-1} \sim B((L_{i-1}+1)K-1, p_i)$. The quantity $\mathbf{E}[L_i]$ can now be computed as,

$$\begin{aligned} \mathbf{E}[L_i] &= \mathbf{E}[\mathbf{E}[L_i|L_{i-1}]] \\ &= \mathbf{E}[((L_{i-1}+1)K-1)p_i] \quad (4) \\ &= p_i K \mathbf{E}[L_{i-1}] + p_i(K-1), \end{aligned}$$

where (4) is due to the fact that $\mathbf{E}[L_i|L_{i-1}]$ is the mean of the binomial random variable $L_i|L_{i-1}$. The above equation can be solved recursively using the initial condition $\mathbf{E}[L_1] = (K-1)p_1$ to get the following closed form expression:

$$\mathbf{E}[L_i] = \sum_{m=1}^{i} \left[K^{i-m}(K-1) \prod_{j=m}^{i} p_j \right].$$

□

Lemma 3. *The probability of error for the tree decoder p_{tree} is given by,*

$$\begin{aligned} p_{tree} &= 1 - G_{L_{n-1}}(0), \qquad \textit{where} \\ G_{L_{n-1}}(z) &= \prod_{i=0}^{n-2} f_{n-1-i}^{K-1}(z), \\ f_k(z) &= \begin{cases} q_k + p_k f_{k+1}^K(z), & 1 \leq k \leq n-1 \\ z^{\frac{1}{K}}, & k = n, \end{cases} \quad (5) \end{aligned}$$

where p_i, q_i are given in (3).

Proof. The quantity p_{tree}, which denotes the probability that more than one path survives the last stage of tree decoding process can be quantified as $P(L_{n-1} \geq 1)$. To compute this probability, we first derive the probability generating function (PGF) $G_{L_{n-1}}(z)$ of the random variable L_{n-1}. The quantity $G_{L_{n-1}}(z)$ is defined as,

$$\begin{aligned} G_{L_{n-1}}(z) &= \mathbf{E}[z^{L_{n-1}}] \\ &= \sum_{k=0}^{K^{n-1}-1} P(L_{n-1}=k) z^k. \quad (6) \end{aligned}$$

Using the fact that $L_{n-1}|L_{n-2} \sim B((L_{i-2}+1)K-1, p_{n-1})$, the above expression can be computed as,

$$\begin{aligned} G_{L_{n-1}}(z) &= \mathbf{E}[z^{L_{n-1}}] \\ &= \mathbf{E}[\mathbf{E}[z^{L_{n-1}}|L_{n-2}]] \\ &= \mathbf{E}[(q_{n-1}+p_{n-1}z)^{(L_{n-2}+1)K-1}] \quad (7) \\ &= (q_{n-1}+p_{n-1}z)^{K-1} G_{L_{n-2}}((q_{n-1}+p_{n-1}z)^K), \end{aligned}$$

where (7) is because $\mathbf{E}[z^{L_{n-1}}|L_{n-2}]$ is the PGF of the binomial random variable $L_{n-1}|L_{n-2}$. The above equation can

be solved recursively with the initial condition $G_{L_1}(z) = (q_1 + p_1 z)^{K-1}$ to yield a closed form solution for the PGF as,

$$G_{L_{n-1}}(z) = \prod_{i=0}^{n-2} f_{n-1-i}^{K-1}(z), \tag{8}$$

$$\text{where } f_k(z) = \begin{cases} q_k + p_k f_{k+1}^K(z), & 1 \le k \le n-1 \\ z^{\frac{1}{K}}, & k = n. \end{cases}$$

Using (6), the quantity p_{tree} can be finally computed as,

$$\begin{aligned} p_{\text{tree}} = P(L_{n-1} \ge 1) &= 1 - P(L_{n-1} = 0) \\ &= 1 - G_{L_{n-1}}(0), \end{aligned}$$

where the quantity $G_{L_{n-1}}(0)$ can be computed by evaluating (8) at $z = 0$. □

We define the computational complexity C of this decoder as the number of nodes on which parity checks need to be performed.

Lemma 4. *A Closed form expressions for computing the expected computational complexity* $\mathbf{E}[C]$ *is given by,*

$$\mathbf{E}[C] = K\left[n - 1 + \sum_{i=1}^{n-2}\sum_{m=1}^{i}\left[K^{i-m}(K-1)\prod_{j=m}^{i} p_j\right]\right], \tag{9}$$

where p_i, q_i *are given in (3).*

Proof. For each non-leaf node that survives the stage i, parity checks need to be done for all it's K children. Hence, computational complexity and the expected computational complexity can be expressed as,

$$\begin{aligned} C &= K + K\left[\sum_{i=1}^{n-2} L_i + n - 2\right], \\ \mathbf{E}[C] &= K + K\left[\sum_{i=1}^{n-2} \mathbf{E}[L_i] + n - 2\right]. \end{aligned} \tag{10}$$

Substituting (3) into (10) yields the final succinct closed form expression for the expected computational complexity given in (9). □

4.1. Choice of the Parity Length Vector

It can be seen from (5) and (9) that there is a trade-off between probability of decoding failure and the average decoding complexity. For a fixed code rate of the tree encoder (fixed number of total parity bits), allocating more parity bits at the initial stages of the tree would result in a lesser average complexity. However, doing this would result in an increased probability of decoding failure. Similarly, allocating more parity bits at the later stages of the tree would result in lesser error rates, but at the expense of higher computational costs. Hence, parity lengths have to be chosen such that both complexity and the error rate can be handled. Towards this end, We formulate a constrained optimization problem of minimizing the expected complexity subject to the probability of decoding failure being less than a carefully chosen threshold $\varepsilon_{\text{tree}}$. Since the parity lengths are non-negative integers, such a problem would be very difficult to solve and hence, we relax the problem to $(l_1, l_2, \ldots, l_{n-1}) \in \mathbb{R}_+^{n-1}$. Also, we replace the constraint $p_{\text{tree}} \le \varepsilon_{\text{tree}}$ with $\mathbf{E}[L_{n-1}] \le \varepsilon_{\text{tree}}$ for the purpose of mathematical tractability (By Markov's inequality, the quantity $\mathbf{E}[L_{n-1}]$ is an upper bound on p_{tree}). Finally, the optimization framework for the choice of parity lengths is given by,

$$\begin{aligned} \underset{(p_1,p_2,\cdots p_{n-1})}{\text{minimize}} \quad & \mathbf{E}[C] \\ \text{subject to} \quad & \mathbf{E}[L_{n-1}] \le \varepsilon_{\text{tree}}, \\ & \sum_{i=1}^{n-1} \log_2\left(\frac{1}{p_i}\right) = M - B, \\ & p_i \in \left[\frac{1}{2^J}, 1\right] \ \forall\, i \in [1 : n-1]. \end{aligned} \tag{11}$$

The above is a Geometric programming [6] opt. problem and can be solved using any standard convex solver. We choose the parity check lengths as $\hat{l}_i = \left\lceil \log_2\left(\frac{1}{\hat{p}_i}\right)\right\rceil$, $\forall\, i \in [1 : n-1]$, where $(\hat{p}_1, \hat{p}_2, \ldots \hat{p}_{n-1})$ is the solution to the optimization problem.

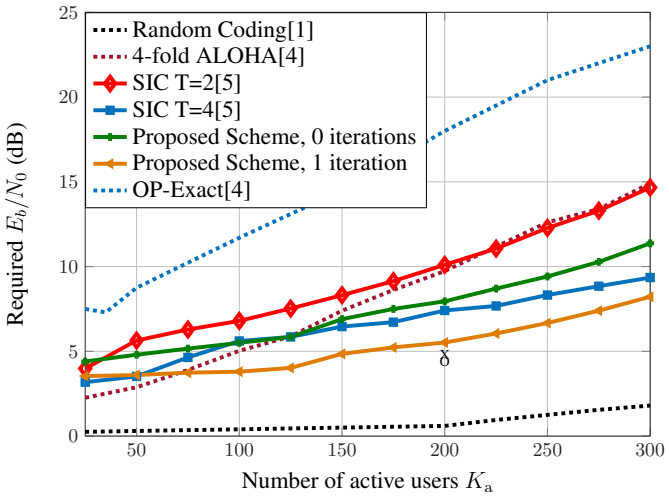

Fig. 2. Minimum E_b/N_0 required to acheive $P_e \le 0.05$ vs. number of users for various schemes. Results for 2 and 3 iterations (see Remark. 1) are represented by 'x' and 'o' respectively. Observe that the SNR gains diminish with each iteration.

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

Table 1. Various parameters used in simulations

5. SIMULATION RESULTS

In this section, we present simulation results to demonstrate the performance of the proposed scheme and provide comparisions with existing schemes in literature. We consider a system with $K_{\mathrm{a}} \in [25 : 300]$ active users, each having $B = 75$ bits of information to transmit. We divide these bits into $n = 11$ sub-blocks and the quantity J, which denotes the length of each sub-block is chosen depending on K_{a} and is given in table (1). Similar to [5], we use sensing matrices that are constructed based on BCH codes for the compressed sensing problem. Specifically, we pick a subset $\mathcal{C}^0$ of codewords of size $|\mathcal{C}^0| = 2^J$ from the (2047,23) BCH codebook $\mathcal{C}$ with the following properties:

(i) $\vec{c} \in \mathcal{C}^0 \implies \vec{1} \oplus \vec{c} \in \mathcal{C} \setminus \mathcal{C}^0$, where $\vec{1} \oplus \vec{c}$ denotes the one's complement of $\vec{c}$.

(ii) $\vec{c}_1, \vec{c}_2 \in \mathcal{C}^0 \implies \vec{c}_1 + \vec{c}_2 \in \mathcal{C}^0$.

(iii) $\vec{0} \in \mathcal{C}^0$, where $\vec{0}$ denotes the all zero codeword.

The subset thus generated has good distance properties ($d_{\text{max}} - d_{\text{min}}$ is very small, where d_{max} and d_{min} denote the maximum and minimum distance between the codewords respectively) and is more suitable for noisy recovery of binary sparse signals than random Gaussian based constructions. We then choose the sensing matrix as $\mathbf{A} = [\vec{a}_0, \vec{a}_1, \cdots, \vec{a}_{2^J-1}]$, of dimension 2047×2^J, where $\vec{a}_i = \sqrt{P}(2\vec{c}_i - 1), \vec{c}_i \in \mathcal{C}^0 \; \forall \; i \in [0 : 2^J - 1]$. The total number of channel uses is then given by $N = 11 \times 2047 = 22,517$. The target error probability of the system is fixed at $\varepsilon = 0.05$. We set list size K for the NNLS CS problem as $K = K_{\mathrm{a}} + 10$. For each $K_{\mathrm{a}} \in [25 : 300]$, we solve the optimization problem (11) using the CVX solver[7] and the resulting solution dictates the choice of parity length vector. Choice of the quantity $\varepsilon_{\text{tree}}$ for each K_{a} is given in table (1). The parameters B and N are chosen such that the rate $\frac{B}{N} = \frac{75}{22,517}$ is approximately the same as the rate resulting due to the choice of parameters $B = 100$ and $N = 30,000$ in [4, 5]. This allows us to make a fair comparison between these schemes and our proposed scheme. We would like to emphasize that the choice of B and N for our simulations is motivated by the existance of good compressive sensing matrices based on BCH codes. When these parameters are proportionally scaled up, performance of the system can only improve, as the finite block length effects are more pronounced for lower values of B and N.

In Fig. 2, E_b/N_0 required to achieve target error probability of 0.05 is plotted as a function of K_{a} for various schemes. The bottom most curve corresponds to Polyanski's acheivability bound[1] on the performance of a finite block length (FBL) code for this model. The curves labelled $T = 2, T = 4$ and 4-fold ALOHA which correspond to the performance curves in [5] and [4] assume the existance of a code for the T-user MAC channel which acheives the bound in [1]. The curve labelled OP-Exact describes the performance of a practical scheme introduced in [4]. It can be seen from Fig. 2 that our proposed scheme after one round of iteration outperforms all the existing schemes for $K_{\mathrm{a}} \in [75 : 300]$. Moreover, similar to the FBL bound, the slope of the curve corresponding to one iteration is very small for small values of K_{a}. We also provide simulation results for 2 and 3 iterations (see Remark. 1) of our proposed scheme with $K_{\mathrm{a}} = 200$ represented by 'x' and 'o' marks respectively. It can be seen that the SNR gains diminish with each iteration.

6. REFERENCES

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

[2] Enrico Paolini, Cedomir Stefanovic, Gianluigi Liva, and Petar Popovski, "Coded random access: applying codes on graphs to design random access protocols," *IEEE Communications Magazine*, vol. 53, no. 6, pp. 144–150, 2015.

[3] Xu Chen, Tsung-Yi Chen, and Dongning Guo, "Capacity of gaussian many-access channels," *IEEE Transactions on Information Theory*, vol. 63, no. 6, pp. 3516–3539, 2017.

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

[5] Avinash Vem, Krishna. R Narayanan, Jun Cheng, and Jean-Francois. Chamberland, "A user-independent serial interference cancellation based coding scheme for the unsourced random access Gaussian channel," in `https://avinashvem.github.io/unsourcedma.pdf`.

[6] Stephen Boyd and Lieven Vandenberghe, *Convex optimization*, Cambridge university press, 2004.

[7] Michael Grant, Stephen Boyd, and Yinyu Ye, "CVX: Matlab software for disciplined convex programming," 2008.