Soft Maximum Likelihood Decoding using GRAND

Amit Solomon
RLE, MIT
Cambridge, MA 02139, USA
amitsol@mit.edu

Ken R. Duffy
Hamilton Institute
Maynooth University, Ireland
ken.duffy@mu.ie

Muriel Médard
RLE, MIT
Cambridge, MA 02139, USA
medard@mit.edu

***Abstract*—Maximum Likelihood (ML) decoding of forward error correction codes is known to be optimally accurate, but is not used in practice as it proves too challenging to efficiently implement. Here we propose a development of a previously described hard detection ML decoder called Guessing Random Additive Noise Decoding (GRAND). We introduce Soft GRAND (SGRAND), a ML decoder that fully avails of soft detection information and is suitable for use with any arbitrary high-rate, short-length block code. We assess SGRAND's performance on Cyclic Redundancy Check (CRC)-aided Polar (CA-Polar) codes, which will be used for all control channel communication in 5G New Radio (NR), comparing its accuracy with CRC-Aided Successive Cancellation List decoding (CA-SCL), a state-of-the-art soft-information decoder specific to CA-Polar codes.**

***Index Terms*—ML decoding, GRAND, 5G NR, CA-Polar**

I. Introduction

Since the work of Shannon [1], Maximum likelihood (ML) decoders have been sought. However, it was established in the 1970s that ML decoding of arbitrary linear codes is an NP-complete problem [2]. Consequently, either codes are co-designed and developed with a specific decoder [3], [4], that is often an approximation of a ML decoder [5], [6], or practical universal decoding algorithms are designed to approximate a ML decoder [7].

An exception to this paradigm is the recently introduced Guessing Random Additive Noise Decoding (GRAND) [8]. In the hard detection setting with additive noise, GRAND has been formally proven to be a ML decoder that works with any block code construction. GRAND attempts to identify the noise that corrupted the code word, rather than directly determining the transmitted code word itself. A computationally more efficient variant of GRAND is GRAND with ABandonment (GRANDAB), which either finds a ML code word or reports a failure after a pre-determined computational cut-off. A failure report from GRANDAB is equivalent to reporting a channel erasure, which is preferable as erasures require less overhead when corrected by an outer code. Both GRAND and GRANDAB have been mathematically established to be capacity achieving when used with random codes [9].

Incorporating soft information into GRAND and GRANDAB decoding would improve accuracy, but it is unclear how to fully do so. An initial attempt that uses one bit of quantized soft information per channel use, which is similar in spirit to how soft information is generated for use with Chase decoding [10], has been proposed [11]. That results in improved accuracy and reduced decoding complexity, but falls short on fully making use of soft information.

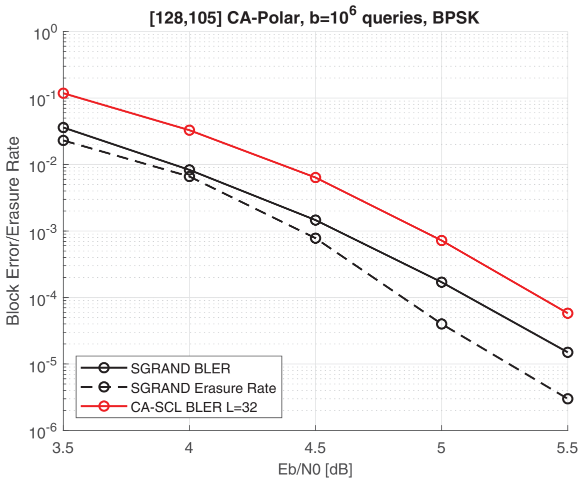


Fig. 1: SGRANDAB and CA-SCL BLER comparison for a [128,105] CA-Polar code. CA-SCL takes list size, L, as an argument. While Matlab defaults to $L = 8$, better BLER performance is seen for higher values and so results for $L = 32$ are presented. Also shown is SGRANDAB's erasure rate with abandonment threshold of $b = 10^6$ code book queries.

In this paper we introduce an advanced variant of GRAND called Soft GRAND (SGRAND) that fully utilizes real-valued channel outputs. We prove that SGRAND is a ML decoder for arbitrary additive memoryless channels, and benchmark its performance by comparison with the state-of-the-art soft detection decoder of 5G New Radio (NR) Cyclic Redundancy Check (CRC)-Aided Polar (CA-Polar) codes, CRC-Aided Successive Cancellation List decoding (CA-SCL) [12], [13], as implemented in the Matlab 5G toolbox. Fig. 1 provides a representative performance evaluation for a [128,105] CA-Polar code where it can be seen that SGRAND with ABandonment (SGRANDAB) obtains better Block Error Rate (BLER), which includes both erroneous decodings and erasures, than CA-SCL. The gap between the two curves at a BLER of 10^{-3} is ≈ 0.5 dB.

The paper is structured as follows. In Section II we discuss the communication model and review relevant background. In Section III we present SGRAND. In Section IV we explain how the simulation was conducted and show additional results. We conclude the paper and discuss future work in Section V.

II. Model and Background

A. Definitions and Notation

Let $x, \vec{x}, X, \vec{X}$ denote a scalar, vector, matrix, and a random vector respectively. All vectors are row vectors, and x_i denotes the i-th element of $\vec{x}$. A linear block code is characterized by a code-length, n, and code-dimension, k, $[n, k]$. $\mathbb{F}_q$ denotes a Galois field with q elements. Composition of functions is denoted by $\circ$, i.e. $f \circ g(x) = f(g(x))$.

B. System model

We consider the standard binary communication setting where an information word $\vec{u} \in \mathbb{F}_2^k$ is encoded with an error correcting code $\alpha : \mathbb{F}_2^k \to \mathbb{F}_2^n$ into a code word $\vec{c} \in \mathbb{F}_2^n$. The code book is the set of all code words $\mathcal{C} = \left\{\vec{c} : \vec{c} = \alpha(\vec{u}), \vec{u} \in \mathbb{F}_2^k\right\}$. For communication on an analog medium, each set of m bits is modulated into a channel input $\vec{x} \in \mathbb{C}^{n/m}$, using a modulation function $\mathbb{F}_2^n \to \mathbb{C}^{n/m}$, that is in turn sent over a noisy channel, where, for convenience, we assume that m divides n. The channel output is denoted by $\vec{Y} = \vec{x} + \vec{Z} \in \mathbb{C}^{n/m}$, where $\vec{Z}$ is random additive noise. A soft decoder is a function $\mathbb{C}^{n/m} \to \mathcal{C}$ that outputs an estimate of $\vec{c}$ from $\vec{y}$. A soft detection ML decoder outputs $\arg\max_{\vec{c} \in \mathcal{C}} p(\vec{y} \mid \vec{c})$, where $p(\vec{y} \mid \vec{c})$ is the likelihood of $\vec{y}$ assuming $\vec{c}$ was transmitted [5, Chapter 1.4].

Each channel output governs the a posteriori probabilities of m demodulated bits, which, as a result of interleaving, are not necessarily sequential. As noise is assumed to impact each transmitted symbol independently, the a posteriori probability of each demodulated bit's value depends on a single channel output. We denote the index of the channel output that determines the a posteriori probability of the i-th bit by $\gamma(i)$.

A demodulator is a function $\theta : \mathbb{C}^{n/m} \to \mathbb{F}_2^n$ that returns the most likely modulated version of the channel output, returning $\theta(\vec{y}) = \arg\max_{\vec{w} \in \mathbb{F}_2^n} p(\vec{y} \mid \vec{w})$. A hard decoder is a function $\mathbb{F}_2^n \to \mathcal{C}$ that receives $\theta(\vec{y}) \in \mathbb{F}_2^n$ and returns a code word.

C. GRAND

GRAND is a hard decoder that identifies the most likely noise sequence, $\vec{z} \in \mathbb{F}_2^n$ as GRAND is a hard decoder, rather than the most likely code word $\vec{c}$. It was shown in [8], [9] that these two problems are equivalent for not-necessarily-memoryless discrete additive channels with uniformly likely input. Identification is achieved by inverting, in order from most likely to least likely based on a probabilistic noise model, the effect of each putative noise sequence from the hard demodulated sequence and querying whether what remains, $\theta(\vec{y}) - \vec{z}$, is in the code book. GRAND only stops at the first instance where the response is affirmative, while GRANDAB additionally halts and reports an erasure if too many code book queries have been made.

Linear codes are the most common forward error correction codes [5], [6]. Testing for code book membership in a linear code book is equivalent to determining if the syndrome of $\theta(\vec{y}) - \vec{z}$ is the zero vector, that is if $H \cdot (\theta(\vec{y}) - \vec{z})^T = \vec{0}^T$, where H is a parity check matrix of the code [5], [6]. Pseudocode for GRANDAB and linear codes is given in Algorithm 1.

Algorithm 1: GRANDAB for a linear code

```
Input: $\theta(\vec{y}), H, b$          ▷ $b$ is max #queries, GRAND: $b = \infty$
Output: $\vec{c}_*$
 1: $g \leftarrow 0$                          ▷ $g$ counts queries performed
 2: while $g \le b$ do
 3:     $\vec{z} \leftarrow$ next most likely noise putative sequence
 4:     $g \leftarrow g + 1$
 5:     if $H \cdot (\theta(\vec{y}) - \vec{z})^T = \vec{0}$ then        ▷ $\mathbb{F}_2$ operations
 6:         $\vec{c}_* \leftarrow \theta(\vec{y}) - \vec{z}$              ▷ ML code word found
 7:         return
 8:     end if
 9: end while
10: $\vec{c}_* \leftarrow \perp$          ▷ Code word not found in $b$ queries; erasure
11: return
```

Note that Algorithm 1 does not specify how to perform the choice of the next putative noise sequence to be considered in line 3. For a Binary Symmetric Channel (BSC), that amounts to generating all sequences of Hamming weight 0, followed by all sequences of Hamming weight 1 and so on, which is a well-studied problem for which efficient recursive algorithms are known [14], [15]. Modelling a well-interleaved communication system, in this paper we establish how to perform GRAND with per-bit soft information for an arbitrary binary memoryless channel.

III. SGRAND

We capture the rank-ordering of demodulated symbols by their reliability via an Ordered Error Indices (OEI) vector $\vec{i} = (i_1, \ldots, i_n)$, which satisfies $p\left(\theta(\vec{y})_{i_j} \mid y_{\gamma(i_j)}\right) \le p\left(\theta(\vec{y})_{i_l} \mid y_{\gamma(i_l)}\right)$ for $j < l$, where $\theta(\vec{y})$ is the most likely demodulated version of $\vec{y}$, and the OEI vector depends on $\vec{y}$, the modulation and the channel statistics, but this dependency is suppressed for notational simplicity. We assume that a priori demodulated probabilities are equal, corresponding to uniform input. The OEI vector is almost surely unique for a continuous memoryless noise distribution.

Pseudocode for SGRANDAB is given in Algorithm 2, which is most readily understood by the example that follows. Suppose $n = 3, m = 1$, and the channel output realizations satisfy: $p\left(\theta(\vec{y})_1 \mid y_{\gamma(1)}\right) = 0.6$, $p\left(\theta(\vec{y})_2 \mid y_{\gamma(2)}\right) = 0.8$, $p\left(\theta(\vec{y})_3 \mid y_{\gamma(3)}\right) = 0.9$, and the probability density function values are $p\left(y_{\gamma(1)}\right) = 0.5$, $p\left(y_{\gamma(2)}\right) = 1$, $p\left(y_{\gamma(3)}\right) = 4$. The unique OEI vector is $(1, 2, 3)$. For the sake of the example, we assume that $H \cdot (\theta(\vec{y}) - \vec{e})^T = \vec{0}^T$ is satisfied only at the last query, identifying an element of the code book. The workings of Algorithm 2 for this setting are given in Table I. We have the following theorem on correctness and progress for SGRAND.

A. Main Theorem

Theorem III.1. *For an additive memoryless channel, Algorithm 2 satisfies the following properties:*

Algorithm 2: SGRANDAB for a linear code

```
Input: $\vec{y}, H, b$                 ▷ $b$ is max #queries, SGRAND: $b = \infty$
Output: $\vec{c}_*$
 1: $g \leftarrow 0$                                   ▷ $g$ counts queries performed
 2: $\mathcal{S} \leftarrow \left\{\vec{0}\right\}$              ▷ $\mathcal{S}$ contains candidate error vectors
 3: $\vec{i} \leftarrow$ OEI vector                                        ▷ Based on $\vec{y}$
 4: while $g \le b$ do
 5:     $\vec{e} \leftarrow \arg\max_{\vec{v} \in \mathcal{S}} p\left(\vec{y} \mid \theta\left(\vec{y}\right) - \vec{v}\right)$
 6:     $\mathcal{S} = \mathcal{S} \setminus \{\vec{e}\}$
 7:     $g \leftarrow g + 1$
 8:     if $H \cdot \left(\theta\left(\vec{y}\right) - \vec{e}\right)^T = \vec{0}^T$ then
 9:         $\vec{c}_* \leftarrow \theta\left(\vec{y}\right) - \vec{e}$                 ▷ ML code word found
10:         return
11:     else                                      ▷ Update $\mathcal{S}$ for the next query
12:         if $\vec{e} = \vec{0}$ then
13:             $j_* \leftarrow 0$
14:         else
15:             $j_* \leftarrow \max\left\{j : e_{i_j} \neq 0\right\}$                      ▷ $j_* > 0$
16:         end if
17:         if $j_* < n$ then
18:             $e_{i(j_*+1)} \leftarrow 1$
19:             $\mathcal{S} = \mathcal{S} \cup \{\vec{e}\}$
20:             if $j_* > 0$ then
21:                 $e_{i_{j_*}} \leftarrow 0$
22:                 $\mathcal{S} = \mathcal{S} \cup \{\vec{e}\}$
23:             end if
24:         end if
25:     end if
26: end while
27: $\vec{c}_* \leftarrow \perp$        ▷ Code word not found in $b$ queries; erasure
28: return
```

g	$\vec{e}$	p	j_*	$\mathcal{S}$
1	$(0,0,0)$	0.432	0	$\{(1,0,0)\}$
2	$(1,0,0)$	0.288	1	$\{(1,1,0),(0,1,0)\}$
3	$(0,1,0)$	0.108	2	$\{(1,1,0),(0,1,1),(0,0,1)\}$
4	$(1,1,0)$	0.072	2	$\{(0,1,1),(0,0,1),(1,1,1),(1,0,1)\}$
5	$(0,0,1)$	0.048	3	$\{(0,1,1),(1,1,1),(1,0,1)\}$
6	$(1,0,1)$	0.032	3	$\{(0,1,1),(1,1,1)\}$
7	$(0,1,1)$	0.012	3	$\{(1,1,1)\}$
8	$(1,1,1)$	0.008	3	$\emptyset$

TABLE I: An example of the execution of Algorithm 2. Column $\vec{e}$ gives the last queried noise sequence. Column $\mathcal{S}$ is the state of the set after each query has been made, at the end of the while loop when all new putative error sequences have been added. The column marked p reports $p\left(\vec{y} \mid \theta\left(\vec{y}\right) - \vec{e}\right)$.

1) ***Correctness:*** *Error vectors are queried in non-increasing order of likelihood, therefore a returned code word is a ML code word.*
2) ***Progress:*** *Each error vector is queried at most once.*

Proof. For an OEI vector $\vec{i}$ we define the *parent* of $\vec{e} \neq \vec{0}$, denoted by $\pi\left(\vec{e}\right)$, as the following vector:

$$\left(\pi\left(\vec{e}\right)\right)_{i_j} = \begin{cases} 0, & \text{if } j = j_*, \\ 1, & \text{if } j = j_* - 1 \\ e_{i_j}, & \text{otherwise} \end{cases}$$

where j_* is defined as in Algorithm 2 and $\pi\left(\vec{0}\right)$ is not defined. We say that $\vec{e}$ is a *child* of $\pi\left(\vec{e}\right)$. While a parent is unique, most, but not all, vectors have two children. The only vectors that do not have two children are: $\vec{0}$, which has a unique child, the vector has 1 in e_{i_1} and zero elsewhere; and those where $j_* = n$, which have no children. For instance, for an OEI vector $\vec{i} = (1,2,3)$, $(1,1,0) = \pi\left((1,1,1)\right) = \pi\left((1,0,1)\right)$. To establish the theorem, first we prove the following lemma.

Lemma III.2. *The following are true for all $\vec{e} \neq \vec{0}$ and an arbitrary memoryless channel:*

1) $\pi\left(\vec{e}\right) \neq \vec{e}$
2) $p\left(\vec{y} \mid \theta\left(\vec{y}\right) - \vec{e}\right) \le p\left(\vec{y} \mid \theta\left(\vec{y}\right) - \pi\left(\vec{e}\right)\right)$
3) $\pi \circ \ldots \circ \pi\left(\vec{e}\right) = \vec{0}$ *after j_* compositions.*

Proof. Let $\vec{e} \neq \vec{0}$, $\vec{i}$ be an OEI vector, and j_* (with respect to $\vec{e}$) be defined as in Algorithm 2.
1) For any such $\vec{e}$, $\left(\vec{e}\right)_{i_{j_*}} = 1, \left(\pi\left(\vec{e}\right)\right)_{i_{j_*}} = 0$, so $\vec{e} \neq \pi\left(\vec{e}\right)$.
2) Let $q_j\left(x\right) = p\left(y_{\gamma(i_j)} \mid \theta\left(\vec{y}\right)_{i_j} - x\right)$, $q\left(\vec{x}\right) = \prod_{j=1}^{n} q_j\left(x_{i_j}\right)$. The existence of j such that $j : q_j\left(e_{i_j}\right) = 0$ completes the proof. Otherwise assume $q\left(\vec{e}\right) > 0$. If $j_* = 1$ or $e_{i_{j_*-1}} = 1$, then $q\left(\vec{e}\right) = q_{j_*}\left(1\right) \prod_{j \neq j_*} q_j\left(e_{i_j}\right)$, and $q\left(\pi\left(\vec{e}\right)\right) = q_{j_*}\left(0\right) \prod_{j \neq j_*} q_j\left(e_{i_j}\right)$, which completes the proof, as $\forall j : q_j\left(1\right) \le q_j\left(0\right)$. Otherwise assume that $j_* > 1, e_{j_*-1} = 0$. Then $q\left(\vec{e}\right) = q_{j_*-1}\left(0\right) q_{j_*}\left(1\right) \prod_{j \neq j_*-1, j_*} q_j\left(e_{i_j}\right)$, and $q\left(\pi\left(\vec{e}\right)\right) = q_{j_*-1}\left(1\right) \cdot q_{j_*}\left(0\right) \cdot \prod_{j \neq j_*} q_j\left(e_{i_j}\right)$. Note that $q\left(\pi\left(\vec{e}\right)\right) / q\left(\vec{e}\right) = q_{j_*-1}\left(1\right) q_{j_*}\left(0\right) / \left(q_{j_*-1}\left(0\right) q_{j_*}\left(1\right)\right) \ge 1$ which completes the proof as j_* comes from an OEI vector.
3) Let $\vec{e} \neq 0, j_* = \max\left\{j : e_{ij} \neq 0\right\}$. By definition of $\pi\left(\vec{e}\right)$, we get $j_* - 1 = \max\left\{j : \left(\pi\left(\vec{e}\right)\right)_{ij} \neq 0\right\}$. By iterating this argument j_* times, we get the zero vector. □

Note that after each query Algorithm 2 adds to $\mathcal{S}$ all the children of the queried noise sequence $\vec{e}$, after removing $\vec{e}$ from $\mathcal{S}$. We return to the proof Theorem III.1 using Lemma III.2. Observe that in order to prove Property 2, it is sufficient to prove that an error vector is added to $\mathcal{S}$ at most once.

We prove properties 1 and 2 by induction on the number of queries g, evaluated at the end of the while loop. The case of $g = 1$ follows immediately as $\mathcal{S}$ is initialized to contain only $\vec{e} = \vec{0}$, which is the most likely error. Then the newly added vector is the unique child of $\vec{0}$, which is not $\vec{0}$, so Property 2 is satisfied. We now assume that the properties are satisfied after g' queries, and establish that they are satisfied after $g' + 1$ queries. For Property 2 suppose by contradiction that after $g' + 1$ queries, a vector $\vec{v}$, that was previously added to $\mathcal{S}$, is added to $\mathcal{S}$. A vector is added to $\mathcal{S}$ only when its parent is queried, so $\vec{v} \neq \vec{0}$, as $\vec{0}$ has no parent. We conclude that at the $g' + 1$ query, the unique parent $\vec{e} = \pi\left(\vec{v}\right)$ is queried, which means that $\vec{e}$ has been added more than once within g' queries, which contradicts the induction assumption.

For Property 1 suppose by contradiction that it is not satisfied, and the next most likely error vector $\vec{v} \notin \mathcal{S}$ that has not been queried before is more likely than $\vec{e}$, i.e. $p(\vec{y} \mid \theta(\vec{y}) - \vec{e}) < p(\vec{y} \mid \theta(\vec{y}) - \vec{v})$, and therefore $\vec{v}$ should have been queried at query number $g' + 1$. We know that $\vec{v} \neq \vec{0}$, as $\vec{0}$ is always the first queried error. $\pi(\vec{v}) \in \mathcal{S}$ cannot hold, as $\pi(\vec{v})$ is at least as likely as $\vec{v}$ due to Lemma III.2, which would contradict $p(\vec{y} \mid \theta(\vec{y}) - \vec{e}) < p(\vec{y} \mid \theta(\vec{y}) - \vec{v})$. Furthermore, if $\pi(\vec{v}) \in \mathcal{S}$ ever held, then $\vec{v}$ would be added to $\mathcal{S}$, which contradicts the assumption that it was never added to $\mathcal{S}$. Since $\pi(\vec{v}) \notin \mathcal{S}$, and has never been, it means that $\pi(\vec{v})$ has not been previously queried. If $p(\vec{y} \mid \theta(\vec{y}) - \vec{v}) < p(\vec{y} \mid \theta(\vec{y}) - \pi(\vec{v}))$, we contradict the assumption that $\vec{v}$ should have been the next query. Otherwise, assume $p(\vec{y} \mid \theta(\vec{y}) - \vec{v}) = p(\vec{y} \mid \theta(\vec{y}) - \pi(\vec{v}))$ and both $\vec{v}, \pi(\vec{v})$ have not been previously queried. By repeating this argument j_* times, we get that $\vec{0}$ has not been queried (see Lemma III.2), which contradicts the fact that it is always the first query of the algorithm. □

B. Complexity considerations

After each iteration of the algorithm, one vector is removed from $\mathcal{S}$, and at most two are added to it. Therefore $\mathcal{S}$ grows by at most one after each query, so after g queries it contains at most $g + 1$ elements. One efficient way of implementing $\mathcal{S}$ is via a Max-Heap [16]. At each iteration Algorithm 2 performs the following: extracts the most likely error vector of $\mathcal{S}$, performs matrix multiplication, and adds at most two elements to $\mathcal{S}$. The complexity of the matrix multiplication, denoted $f(n, k)$, differs across implementations, e.g. [17], [18]. Hence the complexity of the algorithm after g queries, when $\mathcal{S}$ is implemented with a Max-Heap, is $\mathcal{O}(g \cdot f(n,k) \cdot \log g)$. As a result, the worst-case complexity of SGRANDAB with an abandonment threshold of b (as defined in Algorithm 2) is $\mathcal{O}(b \cdot f(n,k) \cdot \log b)$. This is a worst-case analysis, and in practice SGRANDAB often finds a code word much earlier before reaching the abandonment threshold, as can be seen in the simulated results. We also note that the algorithm can be readily parallelized, reducing its time complexity. One mechanism to do so would be to check membership of the code book using different vectors in $\mathcal{S}$ concurrently, while keeping track of each error vector's respective likelihood.

C. SGRAND, no intra-symbol interleaving

In almost all communication systems, and at many layers, bits are interleaved so that correlations in noise that corrupts adjacent bits does not result in correlation at the level of information bits. Interleaving is often performed on bits within the same modulated symbol, as well as across bits from different modulated symbols. While most decoders are designed under assumptions of substantial interleaving, we show how SGRAND can be altered to operate when bits within a modulated symbol are not assumed to be independently impacted by noise. These bits are treated as a symbol in $\mathbb{F}_q$, so that a demodulated symbol is a element of $\mathbb{F}_q, q = 2^m$ and, with a mild abuse of notation, we say that a demodulated vector is $\theta(\vec{y}) \in \mathbb{F}_q^{n/m}$, the most likely demodulated version of $\vec{y}$ over $\mathbb{F}_q^{n/m}$.

In the proof of Theorem III.1 we argued that at each iteration Algorithm 2 removes the queried error sequence $\vec{e}$ from $\mathcal{S}$, performs code book membership test, and adds all the children of $\vec{e}$ to $\mathcal{S}$. The proof relies on error sequences satisfying Lemma III.2, and not on the specific definition of parents and children. As a result, any definition of parents and children that satisfies Lemma III.2 would result in an algorithm that satisfies Theorem III.1.

For the i-th demodulated symbol, we define the *ordered symbol indices* (OSI) vector $\vec{s}^i \in \mathbb{F}_q^q$ as a vector that satisfies the following: 1) $s_1^i = \theta(\vec{y})_i$, 2) $p\left(y_{\gamma(i)} \mid s_j^i\right) \geq p\left(y_{\gamma(i)} \mid s_l^i\right) \geq$ for $j < l$, 3) $\vec{s}^i$ contains each element of $\mathbb{F}_q$ once. An OSI vector depends on $\vec{y}$, the modulation and the statistics of the channel, but this dependency is made implicit for simplicity. We assume that a priori demodulated distribution is uniform over $\mathbb{F}_q$. An OSI vector of the i-th symbol is a vector that rank orders all possible demodulated symbols from most likely to least likely. For the i-th symbol we define $\phi\left(s_j^i\right) = s_{j-1}^i$, and $\phi\left(s_1^i\right)$ is not defined. For example, in the binary case for every bit we have $s_1^i = \theta(\vec{y})_i, s_2^i = 1 - \theta(\vec{y})_i$, $\phi(1 - \theta(\vec{y})_i) = \theta(\vec{y})_i$, where the subtraction is in $\mathbb{F}_2$.

Let $\vec{e} \in \mathbb{F}_q^{n/m}$ be a non-zero error vector, $\vec{i}$ be an OEI vector, j_* be defined as in Algorithm 2, and $\vec{s}^{j_*}$ be an OSI of the i_{j_*}-th symbol. In this scenario we define the *parent* of $\vec{e}$, denoted by $\pi(\vec{e})$, as the following vector:

$$(\pi(\vec{e}))_{i_j} = \begin{cases} \phi\left(e_{i_j}\right), & \text{if } j = j_*, \\ e_{i_j}, & \textit{otherwise} \end{cases}$$

while maintaining the same notation of $\vec{e}$ being a child of $\pi(\vec{e})$. Note that by using this definition, each error has at most n/m children. For example, when $n/m = 4, q = 4$, and assuming the OSI vector of each symbol is $(0, 1, \alpha, \beta)$, for an OEI vector $\vec{i} = (1, 2, 3, 4)$, we have $(1,0,0,0) = \pi((\alpha,0,0,0)) = \pi((1,1,0,0)) = \pi((1,0,1,0)) = \pi((1,0,0,1))$. Lemma III.2 still holds, with the exception of Property 3 requiring at most $(q-1) j_*$ compositions. The proof is similar to the proof given in III-A, and is not given to avoid duplicity. Notice that if we try to define a parent in a similar fashion to the definition of the previous case, i.e. try to change the value of the (i_{j_*-1})-th symbol, we may violate Property 2 of Lemma III.2.

SGRANDAB can be implemented using the new definition of a parent and children, where at each iteration the algorithm adds to $\mathcal{S}$ all the children of the queried error sequence. Theorem III.1 holds in this case, and the proof is similar to the original proof, since Lemma III.2 holds. For similar reasons as before, the worst-case complexity in this scenario is $\mathcal{O}(b \cdot f(n,k) \log(bn/m))$, as at each iteration at most n/m elements added $\mathcal{S}$, while one is removed from $\mathcal{S}$. Thus, unlike most decoding algorithms, SGRAND can be adapted to manage the joint corruption of bits within a symbol by noise.

IV. Performance Evaluation

Polar codes were the first provably capacity-achieving non-random code construction [19], [20]. While they hold promise for reliable communication, poor BLER results were reported based on their initial decoders [12], [13], [21]. Consequently, for all 5G NR control channel communications, CRC is first appended and the whole result Polar coded, resulting in CA-Polar codes.

Existing competitive decoding algorithms use this concatenated structure by first generating a list of possible candidates from the Polar code, and then selecting the best candidate that satisfies the CRC. When combined with soft information, the resulting CA-SCL decoders have shown considerable improvement in BLER performance [12], [13], [22], [23]. As a state-of-the-art implementation of the CA-SCL decoding technique is available in Matlab's 5G toolbox version R2019a [24], we compared SGRANDAB's BLER performance in that environment, relying mostly on Matlab's own code, see Fig. 2.

Note that considered as a single code, CA-Polar codes are themselves linear. They are first encoded with a CRC, interleaved if Downlink, and then encoded with a Polar code. All these operations are linear, hence the encoded message is the result of a linear encoding, $\vec{c} = \vec{u}\, G_{CRC}\, M_{int}\, G_{Polar}$ where G_{CRC}, G_{Polar} are the generator matrices of the CRC code and the Polar code respectively, and M_{int} is the interleaving matrix, which is the identity for Uplink communications. Thus from SGRANDAB's point of view, this is a single linear code and Algorithm 2 can be used for its direct decoding of the concatenated code.

We compared SGRANDAB's performance with Matlab's CA-SCL decoder. While Matlab's default list size for CA-SCL is $L = 8$ [24], also suggested in [25], [26], which is often considered the baseline for 5G performance evaluations [27], at the cost of increased decoding times, we observed enhanced BLER performance with larger list sizes and so report results with $L = 32$, as done for example in [12], [22], [28], [29]. The simulation was conducted on AWGN channel where Gaussian noise with 0 mean and spectral density $\sigma^2 = \mathrm{N}_0/2$ is added independently to each modulated symbol. Here Signal to Noise Ratio (SNR) is defined as $\mathrm{SNR} = -10\log_{10}\left(\sigma^2\right)$ [dB], and $\mathrm{E_b/N_0} = \mathrm{SNR} - 10\log_{10}(k/n) - 10\log_{10}(m)$ [dB], though we note that in Matlab's demo $\mathrm{E_b/N_0} = \mathrm{SNR} - 10\log_{10}\left((k + \mathrm{crc_{len}})/n\right) - 10\log_{10}(m)$, where $\mathrm{crc_{len}}$ is the number of CRC bits, corresponding to the Polar code alone[1].

Setting an abandonment threshold of $b = 10^6$ queries for SGRANDAB, whereupon an erasure is reported, a BLER comparison for the [128,105] Uplink code, subject to Binary Phase Shift Keying (BPSK) modulation, is given in Fig. 1. With the same abandonment threshold as well as with $b = 10^5$, an equivalent result for the [64,46] Uplink code, subject to Quadrature Phase Shift Keying (QPSK) modulation, is shown in Fig. 4a. For the longer code, SGRANDAB outperforms CA-SCL by ≈ 0.5 [dB], and for the shorter code by ≈ 1 [dB], indicating that further performance is available from these codes. In Fig. 4a, for SGRANDAB with $b = 10^6$, errors are not dominated by erasures, so the BLER curve is a good approximation of an unconstrained ML decoder's BLER curve. We see in that case that $b = 10^5$ and $b = 10^6$ have similar BLER curves, indicating that increasing b from 10^5 to 10^6 has little effect on BLER in this scenario.

[1]Matlab's technical support indicates this is to be consistent with 3GPP practice.

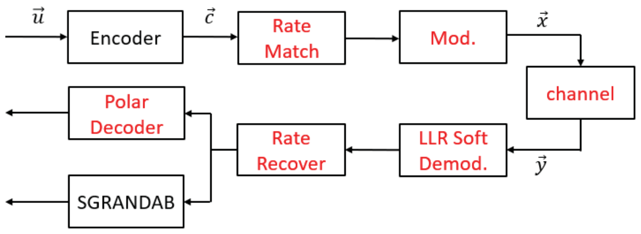


Fig. 2: Simulation overview. Blocks containing red text are used as implemented in Matlab's 5G toolbox.

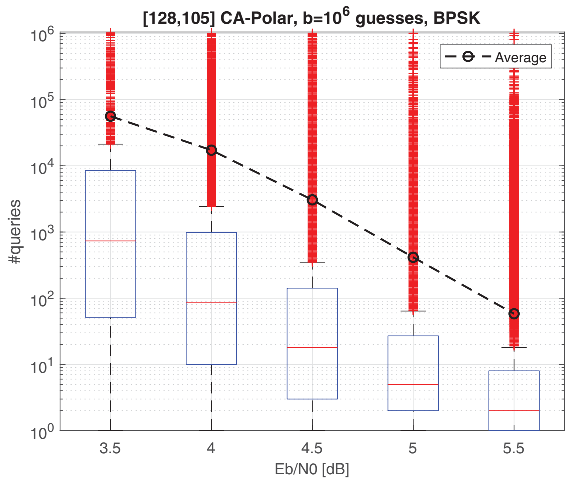
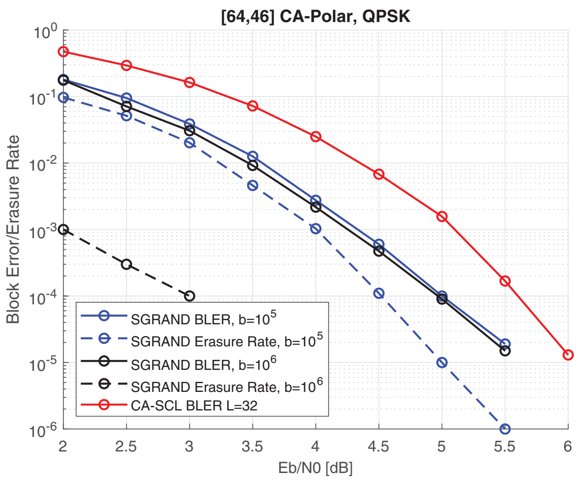


Fig. 3: Number of code book queries until SGRANDAB termination.

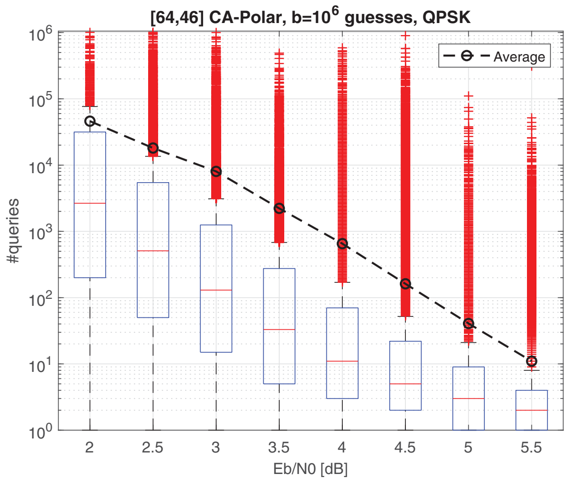
As a proxy for computational complexity, Fig. 3 and Fig. 4b report box plots for the number queries performed until a decoding is found. In all cases, this is significantly lower than the threshold b, which directly affects the complexity of the decoder, as discussed in III-B. In particular, SGRANDAB's complexity gets better as the SNR improves. For comparison, a naïve universal ML decoder that computes the likelihood of each of the 2^k possible code words would require $\approx 10^{13}$, and $\approx 10^{31}$ likelihood calculations for the [64,46] and [128,105] codes, respectively. We also observe that in an AWGN channel, assuming bits are well-interleaved and each bit is independently impacted by a white Gaussian noise, the querying order does not depend on the SNR. The reason for that is that SNR affects the noise's variance, that would result in multiplying the log likelihood ratio of each bit by the same constant, which does not affect the querying order. In other words, assuming an AWGN channel corrupts transmitted code words, SGRANDAB does not require knowledge of the SNR in order to determine the querying order.

V. Conclusion and Discussion

In this paper we introduced SGRAND, a new soft detection ML decoder that is suitable for use with any short-length high-



(a) Block Error Rate.



(b) code book queries until SGRANDAB termination

Fig. 4: Block Error Rate (BLER) comparison of Soft Guessing Random Additive Noise Decoder with Abandonment (SGRANDAB) and Matlab's implementation of [12], [13] in an Additive White Gaussian Noise channel subject to QPSK modulation. b is the abandonment threshold, as defined in Algorithm 2. For further details, see Section IV.

rate block code. For well interleaved channels, we formally proved that when SGRAND returns a code word, it is a ML code word and hence no other decoder can be more accurate. We investigated its performance on 5G NR CA-Polar codes, which will be used for control channel communications in 5G NR, and compared its performance with a state-of-the-art soft CA-SCL decoder, establishing that an extra $0.5 - 1$ [dB] gain is possible through SGRANDAB's use.

While we have introduced SGRAND for memoryless channels, further generalisations of the approach are possible, but are not included here due to space constraints. One variant of SGRAND that can readily be developed is for use with a Markovian channel model, assuming the state of the channel is known.

References

[1] C. E. Shannon, "A mathematical theory of communication," *Bell Labs Tech. J.*, vol. 27, no. 3, pp. 379–423, 1948.

[2] E. Berlekamp, R. McEliece, and H. Van Tilborg, "On the inherent intractability of certain coding problems (corresp.)," *IEEE Tran. Inf. Theory*, vol. 24, no. 3, pp. 384–386, 1978.

[3] J. Conway and N. Sloane, "Soft decoding techniques for codes and lattices, including the golay code and the leech lattice," *IEEE Trans. Inf. Theory*, vol. 32, no. 1, pp. 41–50, 1986.

[4] J. Wolf, "Efficient maximum likelihood decoding of linear block codes using a trellis," *IEEE Trans. Inf. Theory*, vol. 24, no. 1, pp. 76–80, 1978.

[5] R. Roth, *Introduction to coding theory*. CUP, 2006.

[6] S. Lin and D. J. Costello, *Error control coding*. Prentice hall, 2001, vol. 2.

[7] M. P. Fossorier and S. Lin, "Soft-decision decoding of linear block codes based on ordered statistics," *IEEE Transactions on Information Theory*, vol. 41, no. 5, pp. 1379–1396, 1995.

[8] K. R. Duffy, J. Li, and M. Médard, "Guessing noise, not code-words," in *IEEE Int. Symp. Inf. Theory*, 2018, pp. 671–675.

[9] ——, "Capacity-achieving Guessing Random Additive Noise Decoding," *IEEE Tran. Inf. Theory*, vol. 65, no. 7, pp. 4023–4040, 2019.

[10] D. Chase, "A class of algorithms for decoding block codes with channel measurement information," *IEEE Tran. Inf. Theory*, vol. 18, no. 1, pp. 170–182, 1972.

[11] K. R. Duffy and M. Médard, "Guessing random additive noise decoding with soft detection symbol reliability information," in *IEEE Int. Symp. Inf. Theory*, 2019, pp. 480–484.

[12] I. Tal and A. Vardy, "List decoding of polar codes," in *IEEE Int. Symp. Inf. Theory*, 2011, pp. 1–5.

[13] ——, "List decoding of polar codes," *IEEE Tran. Inf. Theory*, vol. 61, no. 5, pp. 2213–2226, 2015.

[14] F. Ruskey and A. Williams, "The coolest way to generate combinations," *Discrete Math.*, vol. 309, no. 17, pp. 5305–5320, 2009.

[15] T. Mazurkiewicz, "An efficient hardware implementation of a combinations generator," *Technical Sciences/University of Warmia and Mazury in Olsztyn*, 2017.

[16] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to algorithms*. MIT press, 2009.

[17] P. Bürgisser, M. Clausen, and M. A. Shokrollahi, *Algebraic complexity theory*. Springer Science & Business Media, 2013, vol. 315.

[18] M. Frigo, C. E. Leiserson, H. Prokop, and S. Ramachandran, "Cache-oblivious algorithms," in *40th Annual Symposium on Foundations of Computer Science (Cat. No. 99CB37039)*. IEEE, 1999, pp. 285–297.

[19] E. Arikan, "Channel polarization: A method for constructing capacity-achieving codes," in *IEEE Int. Symp. Inf. Theory*, 2008, pp. 1173–1177.

[20] ——, "Channel polarization: A method for constructing capacity-achieving codes for symmetric binary-input memoryless channels," *IEEE Tran. Inf. Theory*, vol. 55, no. 7, pp. 3051–3073, 2009.

[21] H. D. Pfister, "A brief introduction to Polar codes," *Supplemental Material for Advanced Channel Coding*, 2014.

[22] K. Niu and K. Chen, "CRC-aided decoding of polar codes," *IEEE Commun. Letters*, vol. 16, no. 10, pp. 1668–1671, 2012.

[23] A. Balatsoukas-Stimming, M. B. Parizi, and A. Burg, "LLR-based successive cancellation list decoding of Polar codes," *IEEE Trans. Signal Process.*, vol. 63, no. 19, pp. 5165–5179, 2015.

[24] "5G New Radio Polar Coding." https://www.mathworks.com/help/releases/R2019a/5g/gs/polar-coding.html, [Online; accessed 04-September-2019].

[25] "3GPP Compliant Polar Encoding/Decoding Chain." https://cdn2.hubspot.net/hubfs/4487277/Assets/Datasheets/AccelerComm_3GPP_Compliant_Polar_EnDecoding_Chain (1).pdf, [Online; accessed 09-October-2019].

[26] L. Xiang, Z. B. K. Egilmez, R. G. Maunder, and L. Hanzo, "CRC-Aided Logarithmic Stack Decoding of Polar Codes for Ultra Reliable Low Latency Communication in 3GPP New Radio," *IEEE Access*, vol. 7, pp. 28 559–28 573, 2019.

[27] V. Bioglio, C. Condo, and I. Land, "Design of Polar codes in 5G new radio," *preprint arXiv:1804.04389*, 2018.

[28] G. Sarkis, P. Giard, A. Vardy, C. Thibeault, and W. J. Gross, "Fast list decoders for polar codes," *IEEE J. Sel. Areas Commun.*, vol. 34, no. 2, pp. 318–328, 2015.

[29] T. Wang, D. Qu, and T. Jiang, "Parity-check-concatenated polar codes," *IEEE Commun. Lett.*, vol. 20, no. 12, pp. 2342–2345, 2016.