

A Random Coding Approach to Performance Analysis of the Locally Constrained Ordered Statistic Decoding

Jifan Liang and Xiao Ma

Abstract

This paper is concerned with the locally constrained ordered statistic decoding (LC-OSD) of binary linear block codes, which is a near maximum-likelihood decoding algorithm. Compared with the conventional OSD, the LC-OSD significantly reduces both the maximum and average number of searches. The former is achieved by performing the serial list Viterbi algorithm (SLVA) over a trellis for local constraints on the test error patterns, while the latter is achieved by incorporating tailored early termination criteria. The main objective of this paper is to explore the relationship between the performance of the LC-OSD and decoding parameters, such as the constraint degree and the maximum list size. To this end, we approximate the local parity-check matrix as a totally random matrix and then estimate the performance of the LC-OSD by analyzing the performance of random codes over the channels associated with the most reliable bits (MRBs). The random coding approach enables us to derive an upper bound on the performance and predict the average rank of the transmitted codeword in

This work was supported in part by the National Key R&D Program of China (No. 2021YFA1000500). (*Corresponding author: Xiao Ma.*)

The authors are with the School of Computer Science and Engineering and the Guangdong Key Laboratory of Information Security Technology, Sun Yat-sen University, Guangzhou 510006, China (e-mail: liangjf56@mail2.sysu.edu.cn; maxiao@mail.sysu.edu.cn).

the list delivered by the LC-OSD. This allows us to balance the constraint degree and the maximum list size for the average (or maximum) time complexity reduction. Simulation results show that the approximation is numerically effective and the random coding approach is powerful. Simulation results also show that the LC-OSD is comparable to other existing decoding algorithms, including soft guessing random additive noise decoding (SGRAND), verifying the efficiency and universality of the LC-OSD.

Index Terms

List size, locally constrained ordered statistic decoding (LC-OSD), performance bounds, random codes, serial list Viterbi algorithm (SLVA).

I. INTRODUCTION

As a central problem in coding theory, the maximum-likelihood decoding (MLD) of a general binary linear code is considered to have no efficient algorithms due to its NP-completeness [1]. The ordered statistics decoding (OSD) investigated by Fossorier and Lin [2] is a near-optimal list decoding algorithm but requires a large number of searches, leading to high complexity. The OSD algorithm first sorts the reliability vector and then selects k linearly independent positions with the most reliabilities, referred to as the most reliable basis (MRB). By re-encoding, the OSD lists all codewords with Hamming distance no greater than t (a preset nonnegative integer) in the MRB and delivers the most likely candidate codeword as output. As a list decoding algorithm, three issues about OSD naturally arise: 1) for what candidate codewords to search, 2) in which order to search, and 3) when to stop the search. To solve these issues, many improvements to OSD have been proposed. For reducing search space, segmentation-discarding OSD (SD-OSD) [3] and linear-equation OSD (LE-OSD) [4] have been proposed; for early stopping, probability-based OSD (PB-OSD) [5] can significantly reduce the number of searches.

Recently, a variant of OSD, called locally constrained OSD (LC-OSD), was proposed further to reduce the decoding complexity [6], [7]. The basic idea of the LC-OSD is to consider an extended subset of reliable positions by including δ more positions next to the MRB, which

was proposed in a report of NASA as early as 1979 [8] and widely used in the information set decoding (ISD) algorithms of cryptography [9]–[11]. The local constraints can be characterized as a trellis specified by a local parity-check matrix, and a list of candidate codewords can be generated by performing the serial list Viterbi algorithm (SLVA) [6]. Aided by tailored early stopping criteria, the LC-OSD can reduce the number of searches to within ten in the high signal-to-noise ratio (SNR) region [7].

The main objective of this paper is to analyze the effects of the decoding parameters on the performance and complexity of the LC-OSD. To this end, we turn to the random coding approach and derive an approximate upper bound on the performance of the LC-OSD and confirm its tightness by simulation. The random coding approach also allows us to approximate the rank of the sent codeword in the search list of the LC-OSD. Armed with these theoretical tools, we suggest a rule to balance the complexity and performance of the LC-OSD.

The rest of this paper is organized as follows. In Sec. II, we illustrate the system model, describe the implementation of the LC-OSD, and analyze the complexity of the LC-OSD. In Sec. III-A, we derive the theoretical results and present some related examples. In Sec. IV, we simulate and compare the performance and complexity of the LC-OSD and various decoding algorithms, including OSD and guessing random additive noise decoding (GRAND), where the results show that the LC-OSD performs better than other decoding algorithms in some cases.

II. LC-OSD

A. System Model

Let $\mathbb{F}_2 = \{0, 1\}$ be the binary field and $\mathcal{C}[n, k]$ be a binary linear code of length n and dimension k . Let \mathbf{G} be a generator matrix of \mathcal{C} and \mathbf{H} be a parity-check matrix of \mathcal{C} . The generator matrix of size $k \times n$ and the parity-check matrix of size $(n - k) \times n$ are related by

$$\mathbf{G}\mathbf{H}^T = \mathbf{O}. \quad (1)$$

Let $\mathbf{u} \in \mathbb{F}_2^k$ be an information vector to be transmitted, which is first encoded into $\mathbf{c} \in \mathbb{F}_2^n$ by

$$\mathbf{c} = \mathbf{u}\mathbf{G}, \quad (2)$$

and then modulated by the binary phase shift keying (BPSK) into a signal vector $\mathbf{x} \in \mathbb{R}^n$ as

$$x_i = (-1)^{c_i}, \quad 1 \leq i \leq n. \quad (3)$$

Then the signal vector \mathbf{x} is transmitted over an AWGN channel, resulting in a received vector $\mathbf{y} \in \mathbb{R}^n$ given by

$$\mathbf{y} = \mathbf{x} + \mathbf{w}, \quad (4)$$

where $\mathbf{w} \sim \text{Normal}(\mathbf{0}, \sigma^2 \mathbf{I}_n)$ is a sample vector of white Gaussian noise. The bit-wise hard-decision vector $\mathbf{z} \in \mathbb{F}_2^n$ is given by

$$z_i \triangleq \begin{cases} 0, & \text{if } y_i \geq 0 \\ 1, & \text{if } y_i < 0 \end{cases}, \quad 1 \leq i \leq n. \quad (5)$$

The log-likelihood ratio (LLR) vector, denoted by $\mathbf{r} \in \mathbb{R}^n$, is defined as

$$r_i \triangleq \log \frac{f_{y|c}(y_i | 0)}{f_{y|c}(y_i | 1)} = \frac{2y_i}{\sigma^2}, \quad 1 \leq i \leq n, \quad (6)$$

where $f_{y|c}(\cdot | \cdot)$ is the conditional probability density function. The reliability of z_i refers to as $|r_i|$. Also, the test error pattern (TEP) $\mathbf{e} \in \mathbb{F}_2^n$ for a test codeword $\mathbf{v} \in \mathbb{F}_2^n$ is given by

$$\mathbf{e} \triangleq \mathbf{z} - \mathbf{v}. \quad (7)$$

The MLD is to find one of the codewords \mathbf{v}^* such that¹

$$\mathbf{v}^* = \arg \max_{\mathbf{v} \in \mathcal{C}} f_{\mathbf{y}|\mathbf{c}}(\mathbf{y} | \mathbf{v}), \quad (8)$$

which is equivalent to

$$\mathbf{v}^* = \arg \min_{\mathbf{v} \in \mathcal{C}} \log \frac{f_{\mathbf{y}|\mathbf{c}}(\mathbf{y} | \mathbf{z})}{f_{\mathbf{y}|\mathbf{c}}(\mathbf{y} | \mathbf{v})}. \quad (9)$$

¹If two or more codewords achieve the maximum, we simply select one at random.

If we define the soft weight² of a TEP e by

$$\Gamma(e) \triangleq \log \frac{f_{\mathbf{y}|c}(\mathbf{y} | \mathbf{z})}{f_{\mathbf{y}|c}(\mathbf{y} | \mathbf{z} - e)} = \sum_{i=1}^n \log \frac{f_{y|c}(y_i | z_i)}{f_{y|c}(y_i | z_i - e_i)} = \sum_{i=1}^n e_i |r_i|, \quad (10)$$

we see that the MLD is equivalent to the lightest-soft-weight decoding. For convenience, we write the soft weight of a TEP e as an inner product. That is,

$$\Gamma(e) = \langle e, |\mathbf{r}| \rangle, \quad (11)$$

where $|\cdot|$ denotes the element-wise absolute value.

B. Locally Constrained OSD

In this subsection, we focus on an arbitrary binary linear code $\mathcal{C}[n, k]$ specified by a parity-check matrix $\mathbf{H} \in \mathbb{F}_2^{(n-k) \times n}$. For a preset constraint degree δ ($0 \leq \delta \leq n - k$) and a maximum number of searches ℓ_{\max} , we restate the LC-OSD algorithm [7] in the following.

- 1) Upon receiving \mathbf{y} , calculate the bit-wise hard-decision vector \mathbf{z} by (5) and the LLR vector \mathbf{r} by (6).
- 2) Sort the LLR vector \mathbf{r} into $\mathbf{r}\mathbf{\Pi}$ such that the first $(n - k - \delta)$ bits of $\mathbf{z}\mathbf{\Pi}$ are linearly independent with least total reliabilities. To be precise, $\mathbf{\Pi}$ is a permutation matrix satisfying:
 - a) the first $(n - k - \delta)$ columns of $\mathbf{H}\mathbf{\Pi}$ is column full rank, i.e., $\text{rank}(\mathbf{H}\mathbf{\Pi})_{[1, n-k-\delta]} = n - k - \delta$, and
 - b) $\sum_{i=1}^{n-k-\delta} |(\mathbf{r}\mathbf{\Pi})_i|$ is as small as possible, where $(\mathbf{r}\mathbf{\Pi})_i$ stands for the i th element of the row vector $(\mathbf{r}\mathbf{\Pi})$.

This can be accomplished by a greedy algorithm (Algorithm 1).

²This is called weighted Hamming distance in [12] and ellipsoidal weight in [13].

Algorithm 1 Find a permutation matrix for OSD

Input: The parity-check matrix $\mathbf{H} \in \mathbb{F}_2^{(n-k) \times n}$, the reliability vector $|\mathbf{r}| \in \mathbb{R}^n$, the constraint degree δ ($0 \leq \delta \leq n - k$).

Output: A permutation matrix $\mathbf{\Pi}$ satisfying the requirements (2a–2b).

```

1: function GET-PERMUTATION( $\mathbf{H}, |\mathbf{r}|, \delta$ )
2:    $\mathbf{\Pi} \leftarrow$  an ascending sorted permutation of vector  $|\mathbf{r}|$ .  $\triangleright$  That is,  $|\mathbf{r}|\mathbf{\Pi}$  is non-decreasing.
3:    $i' \leftarrow n - k - \delta + 1$ .
4:   for  $i = 1, 2, \dots, n - k - \delta$  do
5:     while  $\text{rank}(\mathbf{H}\mathbf{\Pi})_{[1,i]} < i$  do  $\triangleright (\mathbf{H}\mathbf{\Pi})_{[1,i]}$  is formed by the first  $i$  columns of  $\mathbf{H}\mathbf{\Pi}$ .
6:        $\text{swap}(\mathbf{\Pi}_i, \mathbf{\Pi}_{i'})$ .  $\triangleright$  Swap the columns  $i$  and  $i'$  of  $\mathbf{\Pi}$ .
7:        $i' \leftarrow i' + 1$ .
8:   return  $\mathbf{\Pi}$ .
```

For convenience, we denote the permuted \mathbf{z} by $\tilde{\mathbf{z}}$, i.e.,

$$\tilde{\mathbf{z}} \triangleq \mathbf{z}\mathbf{\Pi}, \quad (12)$$

and use similar notations for vectors $\mathbf{r}, \mathbf{e}, \mathbf{v}$ etc.

3) Perform the Gaussian elimination for $\mathbf{H}\mathbf{\Pi}$, resulting in $\tilde{\mathbf{H}} = \mathbf{Q}\mathbf{H}\mathbf{\Pi}$ of form

$$\tilde{\mathbf{H}} = \left[\begin{array}{c|c} \begin{array}{c} n-k-\delta \text{ columns} \\ \mathbf{I} \end{array} & \begin{array}{c} k+\delta \text{ columns} \\ \mathbf{P}_1 \end{array} \\ \hline \begin{array}{c} \mathbf{O} \end{array} & \begin{array}{c} \mathbf{P}_2 \end{array} \end{array} \right] \begin{array}{l} n-k-\delta \text{ rows} \\ \delta \text{ rows} \end{array} \quad (13)$$

where \mathbf{I} denotes the identity matrix of order $n - k - \delta$.

4) A test vector $\mathbf{v} = \mathbf{z} - \mathbf{e}$ is a codeword of \mathcal{C} if and only if the TEP \mathbf{e} satisfies the

parity-check equation

$$\mathbf{H}\mathbf{v}^T = \mathbf{H}(\mathbf{z}^T - \mathbf{e}^T) = \mathbf{0}, \quad (14)$$

or equivalently, in the permuted form,

$$\tilde{\mathbf{H}}\tilde{\mathbf{v}}^T = \tilde{\mathbf{H}}(\tilde{\mathbf{z}}^T - \tilde{\mathbf{e}}^T) = \mathbf{0}. \quad (15)$$

So, searching a (permuted) codeword $\tilde{\mathbf{v}}$ is equivalent to searching a TEP $\tilde{\mathbf{e}}$ because $\tilde{\mathbf{z}}$ is fixed upon receiving \mathbf{y} . If we divide $\tilde{\mathbf{e}}$ (and $\tilde{\mathbf{z}}$) into two parts of lengths $(n - k - \delta)$ and $(k + \delta)$, namely

$$\tilde{\mathbf{e}} = \left[\begin{array}{c|c} \overset{n-k-\delta \text{ bits}}{\tilde{\mathbf{e}}_L} & \overset{k+\delta \text{ bits}}{\tilde{\mathbf{e}}_R} \end{array} \right], \quad \tilde{\mathbf{z}} = \left[\begin{array}{c|c} \overset{n-k-\delta \text{ bits}}{\tilde{\mathbf{z}}_L} & \overset{k+\delta \text{ bits}}{\tilde{\mathbf{z}}_R} \end{array} \right], \quad (16)$$

then by expanding (15), we can obtain equations with unknowns $\tilde{\mathbf{e}} = (\tilde{\mathbf{e}}_L, \tilde{\mathbf{e}}_R)$,

$$\tilde{\mathbf{e}}_L^T + \mathbf{P}_1 \tilde{\mathbf{e}}_R^T = \tilde{\mathbf{s}}_1, \quad (17)$$

$$\mathbf{P}_2 \tilde{\mathbf{e}}_R^T = \tilde{\mathbf{s}}_2, \quad (18)$$

where the right-hand sides (RHSs) of (17) and (18) are given by

$$\tilde{\mathbf{s}}_1 \triangleq \tilde{\mathbf{z}}_L^T + \mathbf{P}_1 \tilde{\mathbf{z}}_R^T, \quad (19)$$

$$\tilde{\mathbf{s}}_2 \triangleq \mathbf{P}_2 \tilde{\mathbf{z}}_R^T, \quad (20)$$

which will be calculated and stored before searching codewords to avoid replicated computations.

As in (10), the soft weight of $\tilde{\mathbf{e}}$ is defined as³

$$\Gamma(\tilde{\mathbf{e}}) \triangleq \langle \tilde{\mathbf{e}}, |\tilde{\mathbf{r}}| \rangle. \quad (21)$$

Similarly, define

$$\Gamma(\tilde{\mathbf{e}}_L) \triangleq \langle \tilde{\mathbf{e}}_L, |\tilde{\mathbf{r}}_L| \rangle, \Gamma(\tilde{\mathbf{e}}_R) \triangleq \langle \tilde{\mathbf{e}}_R, |\tilde{\mathbf{r}}_R| \rangle. \quad (22)$$

³Strictly speaking, it should use $\tilde{\Gamma}(\tilde{\mathbf{e}})$ instead of $\Gamma(\tilde{\mathbf{e}})$ for $\tilde{\mathbf{r}}$.

We have

$$\Gamma(\tilde{\mathbf{e}}) = \Gamma(\tilde{\mathbf{e}}_{\text{L}}) + \Gamma(\tilde{\mathbf{e}}_{\text{R}}). \quad (23)$$

- 5) From (17), we see that $\tilde{\mathbf{e}}_{\text{L}}$ (and $\tilde{\mathbf{e}}$ by (16)) are uniquely determined by $\tilde{\mathbf{e}}_{\text{R}}$. Hence, we only need to search for $\tilde{\mathbf{e}}_{\text{R}}$ instead of $\tilde{\mathbf{e}}$ (or $\tilde{\mathbf{v}}$). Intuitively, we should make a list of $\tilde{\mathbf{e}}_{\text{R}}$

$$\tilde{\mathbf{e}}_{\text{R}}^{(1)}, \tilde{\mathbf{e}}_{\text{R}}^{(2)}, \dots, \tilde{\mathbf{e}}_{\text{R}}^{(\ell_{\max})} \quad (24)$$

satisfying (18) in an order such that the partial soft weights are non-decreasing, i.e.,

$$\Gamma(\tilde{\mathbf{e}}_{\text{R}}^{(1)}) \leq \Gamma(\tilde{\mathbf{e}}_{\text{R}}^{(2)}) \leq \dots \leq \Gamma(\tilde{\mathbf{e}}_{\text{R}}^{(\ell_{\max})}). \quad (25)$$

This can be achieved by the SLVA⁴ (Algorithm 4 in Appendix D) over a trellis specified by the local parity-check matrix \mathbf{P}_2 .

- 6) For each $\tilde{\mathbf{e}}_{\text{R}}^{(j)}$ ($1 \leq j \leq \ell_{\max}$), solve for $\tilde{\mathbf{e}}^{(j)}$ by (17) and (16).
 7) Finally, output the codeword corresponding to one of the $\tilde{\mathbf{e}}^{(j)}$ with the lightest soft weight.

For the sake of clarity, the LC-OSD is summarized in Algorithm 2 with an optional early stopping criterion (see Sec. III-A for details).

⁴The basic idea of the SLVA was initially presented by Seshadri and Sundberg in [14]. The description in Appendix D is based on Wenchao Lin's thesis [15], which is believed to be more general than that given in [14].

Algorithm 2 Locally constrained OSD**Input:** \mathbf{H} , \mathbf{y} , δ , ℓ_{\max} , an early stopping criterion Ψ (optional).**Output:** The optimal searched codeword \mathbf{v} .

```

1: function LC-OSD( $\mathbf{H}$ ,  $\mathbf{y}$ ,  $\delta$ ,  $\ell_{\max}$ ,  $\Psi$ )
2:    $\mathbf{z} \leftarrow$  the bit-wise hard decision vector of  $\mathbf{y}$ .  $\triangleright$  Calculate by (5).
3:    $\mathbf{r} \leftarrow$  the reliability vector of  $\mathbf{z}$ .  $\triangleright$  Calculate by (6).
4:    $\mathbf{\Pi} \leftarrow$  GET-PERMUTATION( $\mathbf{H}$ ,  $\mathbf{r}$ ,  $\delta$ ).  $\triangleright$  Get a permutation matrix.
5:   Permute  $\mathbf{z}$ ,  $\mathbf{r}$  into  $\tilde{\mathbf{z}}$ ,  $\tilde{\mathbf{r}}$  by  $\mathbf{\Pi}$ .  $\triangleright$  Permute as (12).
6:    $\tilde{\mathbf{H}} \leftarrow$  Gaussian eliminated matrix of  $\mathbf{H}\mathbf{\Pi}$ .  $\triangleright$  Eliminate into the form as (13).
7:   Extract the sub-matrices  $\mathbf{P}_1$  and  $\mathbf{P}_2$  from  $\tilde{\mathbf{H}}$ .  $\triangleright$  Extract as (13).
8:    $\tilde{\mathbf{s}}_1 \leftarrow \tilde{\mathbf{z}}_L^T + \mathbf{P}_1 \tilde{\mathbf{z}}_R^T$ .  $\triangleright$  Calculate the RHS of (19).
9:    $\tilde{\mathbf{s}}_2 \leftarrow \mathbf{P}_2 \tilde{\mathbf{z}}_R^T$ .  $\triangleright$  Calculate the RHS of (20).
10:   $\tilde{\mathbf{e}}_{\text{opt}} \leftarrow \tilde{\mathbf{z}}$ .  $\triangleright$  Note that  $\tilde{\mathbf{z}}$  is a valid TEP for test.
11:  for  $\ell = 1, 2, \dots, \ell_{\max}$  do
12:     $\tilde{\mathbf{e}}_R^{(\ell)} \leftarrow \text{SLVA}(\mathbf{P}_2, |\tilde{\mathbf{r}}_R|, \tilde{\mathbf{s}}_2, \ell)$ .  $\triangleright$  Search for the  $\ell$ th lightest TEP.
13:     $\tilde{\mathbf{e}}_L^{(\ell)} \leftarrow \tilde{\mathbf{s}}_1 - \mathbf{P}_1(\tilde{\mathbf{e}}_R^{(\ell)})^T$ .  $\triangleright$  Solve from (17).
14:     $\tilde{\mathbf{e}}^{(\ell)} = (\tilde{\mathbf{e}}_L^{(\ell)}, \tilde{\mathbf{e}}_R^{(\ell)})$ .
15:    if  $\Gamma(\tilde{\mathbf{e}}_{\text{opt}}) > \Gamma(\tilde{\mathbf{e}}^{(\ell)})$  then  $\triangleright$  Update the optimal TEP with LSW.
16:       $\tilde{\mathbf{e}}_{\text{opt}} \leftarrow \tilde{\mathbf{e}}^{(\ell)}$ .
17:    if the early stopping criterion  $\Psi$  is satisfied then  $\triangleright$  See Sec.III-A for details.
18:      break.
19:   $\mathbf{v} \leftarrow \mathbf{z} - \tilde{\mathbf{e}}_{\text{opt}} \mathbf{\Pi}^{-1}$ .
20:  return  $\mathbf{v}$ .

```

Remarks. If $\delta = 0$, Algorithm 2 reduces to the OSD algorithm [2] with a search order the same as that of the flipping pattern tree (FPT) [16]; if $\delta = n - k$, it is indeed the MLD.

C. Complexity Analysis

In this subsection, we focus on the time and space complexity of the LC-OSD (Algorithm 2).

1) *Time Complexity*: We analyze the time complexity of the algorithm by evaluating the number of floating point operations (FLOPs) or binary operations (BOPs) required by each step. The time complexity consists of three dominant parts:

- 1) *Gaussian elimination*. Reducing \mathbf{H} into a block upper triangular matrix as shown in (13) requires $\mathcal{O}((n-k)(n-k-\delta))$ elementary row operations, each taking $\mathcal{O}(n)$ BOPs.
- 2) *Initializing the SLVA*. The trellis (see Fig. 12 for reference) specified by the local parity-check matrix \mathbf{P}_2 has $(k+\delta)$ sections, and each level has at most 2^δ states. To find the best path $\tilde{\mathbf{e}}_{\mathbf{R}}^{(1)}$, the SLVA needs to calculate and store the best paths associated with all allowable states, each requiring $\mathcal{O}(1)$ BOPs and FLOPs.
- 3) *Re-encoding*. With initializations done, searching a candidate $\tilde{\mathbf{e}}_{\mathbf{R}}^{(j)}$ ($j > 1$) by the SLVA (line 12, Algorithm 2) requires $\mathcal{O}(n)$ FLOPs and re-encoding $\tilde{\mathbf{e}}_{\mathbf{L}}^{(j)}$ by (17) (line 13, Algorithm 2) requires $\mathcal{O}((n-k-\delta)(k+\delta))$ BOPs (recalling that \mathbf{P}_1 is of size $(n-k-\delta) \times (k+\delta)$).

To summarize, the average time complexity is

$$T_{\text{avg}} = \underbrace{\mathcal{O}(n(n-k)(n-k-\delta))}_{\text{Gaussian elimination}} + \underbrace{\mathcal{O}(2^\delta(k+\delta))}_{\text{initializing the SLVA}} + \underbrace{\ell_{\text{avg}}\mathcal{O}((n-k-\delta)(k+\delta))}_{\text{re-encoding}}, \quad (26)$$

where ℓ_{avg} denotes the average number⁵ of searches per frame, and the worst-case time complexity is

$$T_{\text{max}} = \mathcal{O}(n(n-k)(n-k-\delta)) + \mathcal{O}(2^\delta(k+\delta)) + \ell_{\text{max}}\mathcal{O}((n-k-\delta)(k+\delta)). \quad (27)$$

2) *Space Complexity*: We analyze the space complexity of the algorithm by calculating the number of bytes used. The space complexity consists of two dominant parts:

- 1) *Matrices and vectors*. The algorithm needs to store matrices \mathbf{H} , \mathbf{P}_1 and \mathbf{P}_2 of size $\mathcal{O}((n-k) \times n)$ and several vectors of length $\mathcal{O}(n)$.

⁵Usually, ℓ_{avg} is less than ℓ_{max} because of the early stopping criterion (Sec. III-A).

- 2) *The SLVA and search.* The main storage space of SLVA includes the space occupied by the initialization, $\mathcal{O}(2^\delta(k + \delta))$ and the space occupied by searching and storing paths, $\mathcal{O}(\ell_{\max}(k + \delta))$.

In summary, the space complexity is given by

$$S_{\max} = \underbrace{\mathcal{O}(n(n - k))}_{\text{matrices and vectors}} + \underbrace{\mathcal{O}((2^\delta + \ell_{\max})(k + \delta))}_{\text{the SLVA}}. \quad (28)$$

III. PERFORMANCE ANALYSIS OF THE LC-OSD

A. Early Stopping Criteria

The complexity analysis in Sec. II-C suggests that the complexity can be reduced by limiting the maximum number of tests along with properly designed early stopping criteria. Let $\tilde{\mathbf{e}}^{(\ell)} = (\tilde{\mathbf{e}}_{\text{L}}^{(\ell)}, \tilde{\mathbf{e}}_{\text{R}}^{(\ell)})$ be the ℓ th searched TEP in the LC-OSD algorithm (Algorithm 2) and denote by $\tilde{\mathbf{e}}_{\text{opt}}^{(j)}$ the up-to-date optimal TEP after the j th iteration. That is,

$$\tilde{\mathbf{e}}_{\text{opt}}^{(j)} = \arg \min_{\tilde{\mathbf{e}}^{(\ell)}: 1 \leq \ell \leq j} \Gamma(\tilde{\mathbf{e}}^{(\ell)}). \quad (29)$$

Example 1. To illustrate the relationship among

$$\tilde{\mathbf{e}}_{\text{R}}^{(\ell)}, \tilde{\mathbf{e}}_{\text{L}}^{(\ell)}, \tilde{\mathbf{e}}^{(\ell)}, \tilde{\mathbf{e}}_{\text{opt}}^{(\ell)}, \quad (30)$$

we have plotted their soft weights by simulating on a $\mathcal{C}[128, 64]$ code with $E_b/N_0 = 2.5$ dB.

As seen from Fig. 1, we can prove (omitted here) that, for all ℓ ,

- $\Gamma(\tilde{\mathbf{e}}^{(\ell)}) = \Gamma(\tilde{\mathbf{e}}_{\text{L}}^{(\ell)}) + \Gamma(\tilde{\mathbf{e}}_{\text{R}}^{(\ell)})$ (by definition),
- $\Gamma(\tilde{\mathbf{e}}_{\text{R}}^{(\ell)})$ is non-decreasing (due to the nature of the SLVA),
- $\Gamma(\tilde{\mathbf{e}}_{\text{opt}}^{(\ell)})$ is non-increasing (by definition), and
- $\Gamma(\tilde{\mathbf{e}}_{\text{opt}}^{(\ell)}) \leq \Gamma(\tilde{\mathbf{e}}^{(\ell)})$ (by definition).

Another trivial fact is that, if

$$\Gamma(\tilde{\mathbf{e}}_{\text{opt}}^{(j)}) \leq \Gamma(\tilde{\mathbf{e}}_{\text{R}}^{(j)}) \quad (31)$$

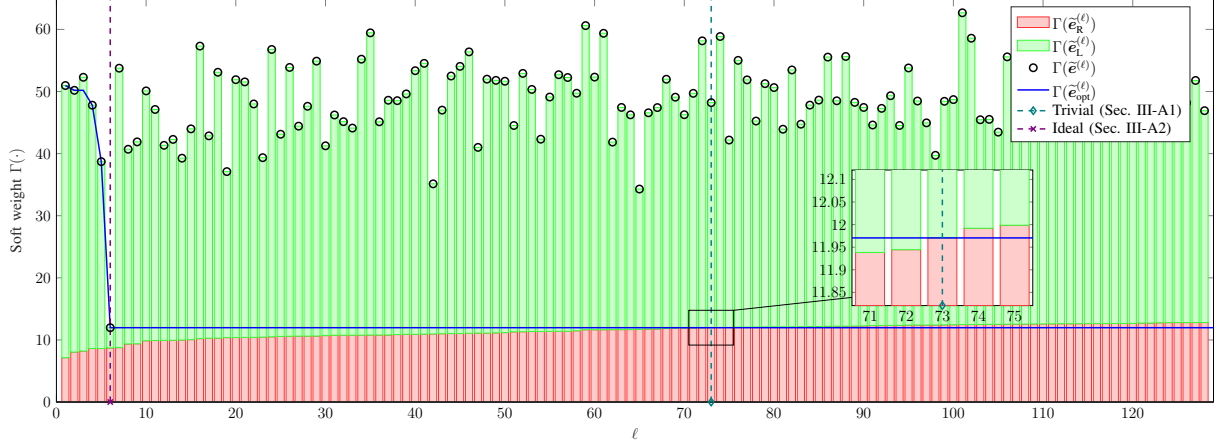


Fig. 1. Numerical illustration of the relationships among soft weights $\Gamma(\cdot)$. The data are obtained by simulating a $\mathcal{C}[128, 64]$ code with $E_b/N_0 = 2.5$ dB, where the LC-OSD parameters are $(\delta, \ell_{\max}) = (8, 2^7)$.

holds for some j ($1 \leq j \leq \ell_{\max}$), then it also holds for any $j' > j$. Based on this fact, we propose the following stopping criterion.

1) *Trivial Stopping Criterion:* Before stating the criterion, we present an obvious proposition.

Proposition 1. Let ℓ_{\max} be a positive integer. If, for some j ($1 \leq j \leq \ell_{\max}$),

$$\Gamma(\tilde{\mathbf{e}}_{\text{opt}}^{(j)}) \leq \Gamma(\tilde{\mathbf{e}}^{(\ell)}) \quad (32)$$

holds for all ℓ ($j < \ell \leq \ell_{\max}$), then $\tilde{\mathbf{e}}_{\text{opt}}^{(j)}$ is one of the lightest TEPs in the search list, i.e.,

$$\tilde{\mathbf{e}}_{\text{opt}}^{(j)} = \arg \min_{\tilde{\mathbf{e}}^{(\ell)}: 1 \leq \ell \leq \ell_{\max}} \Gamma(\tilde{\mathbf{e}}^{(\ell)}). \quad (33)$$

Proof: By definition (29),

$$\Gamma(\tilde{\mathbf{e}}_{\text{opt}}^{(j)}) \leq \Gamma(\tilde{\mathbf{e}}^{(\ell)}) \quad (34)$$

holds for all $\ell \leq j$. Moreover, by assumption, it also holds for all $\ell > j$. Therefore, (34) holds for all ℓ . ■

Immediately, we have the following criterion.

Trivial stopping criterion. If, for some j , $\Gamma(\tilde{\mathbf{e}}_{\text{opt}}^{(j)}) \leq \Gamma(\tilde{\mathbf{e}}_{\text{R}}^{(j)})$, terminate the LC-OSD algorithm and deliver $\mathbf{v} = \mathbf{z} - \tilde{\mathbf{e}}_{\text{opt}}^{(j)}\mathbf{\Pi}^{-1}$ as the output.

Proposition 2. *If the LC-OSD is stopped with the trivial stopping criterion being satisfied (usually requiring a sufficiently large ℓ_{\max}), the output is an ML codeword.*

Proof: For all $\ell \geq j$, we have

$$\Gamma(\tilde{\mathbf{e}}_{\text{opt}}^{(j)}) \leq \Gamma(\tilde{\mathbf{e}}_{\text{R}}^{(j)}) \quad (\text{by assumption}) \quad (35)$$

$$\leq \Gamma(\tilde{\mathbf{e}}_{\text{R}}^{(\ell)}) \quad (\Gamma(\tilde{\mathbf{e}}_{\text{R}}^{(j)}) \text{ is non-decreasing}) \quad (36)$$

$$\leq \Gamma(\tilde{\mathbf{e}}^{(\ell)}) \quad (\text{by (23)}) \quad (37)$$

which satisfies the conditions of Proposition 1. ■

This proposition implies that using the trivial stopping criterion would not incur performance loss. However, as we can see in Fig. 1, the termination raised by the trivial stopping criterion is quite behind the location of the true TEP. So, if we could know $\Gamma(\tilde{\mathbf{e}}_{\text{L}})$ in advance, the LC-OSD can terminate as soon as possible. Based on this idea, we propose several early stopping criteria.

2) *Ideal Stopping Criterion:* Let $\tilde{\mathbf{e}}$ denote the true TEP, i.e.,

$$\tilde{\mathbf{e}} \triangleq \tilde{\mathbf{z}} - \tilde{\mathbf{c}}, \quad (38)$$

where $\tilde{\mathbf{c}} = \mathbf{c}\mathbf{\Pi}$ is the permuted transmitted codeword.

Ideal stopping criterion. Assume that $\Gamma(\tilde{\mathbf{e}}_{\text{L}})$ were known. If for some j , $\Gamma(\tilde{\mathbf{e}}_{\text{opt}}^{(j)}) < \Gamma(\tilde{\mathbf{e}}_{\text{L}}) + \Gamma(\tilde{\mathbf{e}}_{\text{R}}^{(j)})$, terminate the LC-OSD algorithm and deliver $\mathbf{v} = \mathbf{z} - \tilde{\mathbf{e}}_{\text{opt}}^{(j)}\mathbf{\Pi}^{-1}$ as the output.

Proposition 3. *If the LC-OSD were stopped with the ideal stopping criterion being satisfied, the performance of the LC-OSD would be no worse⁶ (or even better) than that of the MLD.*

Proof: When the LC-OSD is stopped by the ideal stopping criterion, there are two cases.

⁶This is not surprising since knowing $\Gamma(\tilde{\mathbf{e}}_{\text{L}})$ in advance is an ideal but impractical assumption.

- The first case is that the true TEP \tilde{e} has already been searched, i.e., $\tilde{e}_R = \tilde{e}_R^{(\ell)}$ for some $\ell \leq j$. In this case, it is not necessary to continue the remaining search.
- The second case is that the true TEP \tilde{e} has not been searched, indicating that $\Gamma(\tilde{e}_R) \geq \Gamma(\tilde{e}_R^{(j)})$. By assumption, it follows that

$$\Gamma(\tilde{e}_{\text{opt}}^{(j)}) < \Gamma(\tilde{e}_L) + \Gamma(\tilde{e}_R^{(j)}) \leq \Gamma(\tilde{e}_L) + \Gamma(\tilde{e}_R) = \Gamma(\tilde{e}), \quad (39)$$

which implies that the MLD must be in error and further search is unnecessary either. ■

The ideal criterion is useless for decoding because we do not know $\Gamma(\tilde{e}_L)$ in advance.

3) *Approximate Ideal Stopping Criteria:* Although the ideal criterion is not helpful in practice, we may replace $\Gamma(\tilde{e}_L)$ by its expectation, as presented in the following.

Dynamic approximate ideal (DAI) stopping criterion. If for some j , $\Gamma(\tilde{e}_{\text{opt}}^{(j)}) \leq \tau_{\text{DAI}} + \Gamma(\tilde{e}_R^{(j)})$, terminate the LC-OSD algorithm and deliver $\mathbf{v} = \mathbf{z} - \tilde{\mathbf{e}}_{\text{opt}}^{(j)} \mathbf{\Pi}^{-1}$ as the output, where

$$\tau_{\text{DAI}} \triangleq \mathbb{E}[\Gamma(\tilde{e}_L) \mid \mathbf{r}] \quad (40)$$

can be calculated as (see Appendix A for details)

$$\tau_{\text{DAI}} = \sum_{i=1}^{n-k-\delta} \frac{1}{1 + \exp(|\tilde{r}_i|)} \cdot |\tilde{r}_i|. \quad (41)$$

By using integral to estimate the RHS of (41), we can get the approximation of $\mathbb{E}[\Gamma(\tilde{e}_L)]$ and the following stopping criterion.

Static approximate ideal (SAI) stopping criterion. If for some j , $\Gamma(\tilde{e}_{\text{opt}}^{(j)}) \leq \tau_{\text{SAI}} + \Gamma(\tilde{e}_R^{(j)})$, terminate the LC-OSD algorithm and deliver $\mathbf{v} = \mathbf{z} - \tilde{\mathbf{e}}_{\text{opt}}^{(j)} \mathbf{\Pi}^{-1}$ as the output, where

$$\tau_{\text{SAI}} \triangleq (n - k - \delta) \mathbb{E}[e|r| \mid |r| \leq \alpha] \quad (42)$$

is the approximation of $\mathbb{E}[\Gamma(\tilde{e}_L)]$ and α is the $(n - k - \delta)$ th n -quantile for the random variable $|r|$. The SAI threshold τ_{SAI} can be calculated as (see Appendix B for details)

$$\tau_{\text{SAI}} = n \int_0^\alpha \frac{r}{1 + \exp(r)} f_{|r|}(r) dr. \quad (43)$$

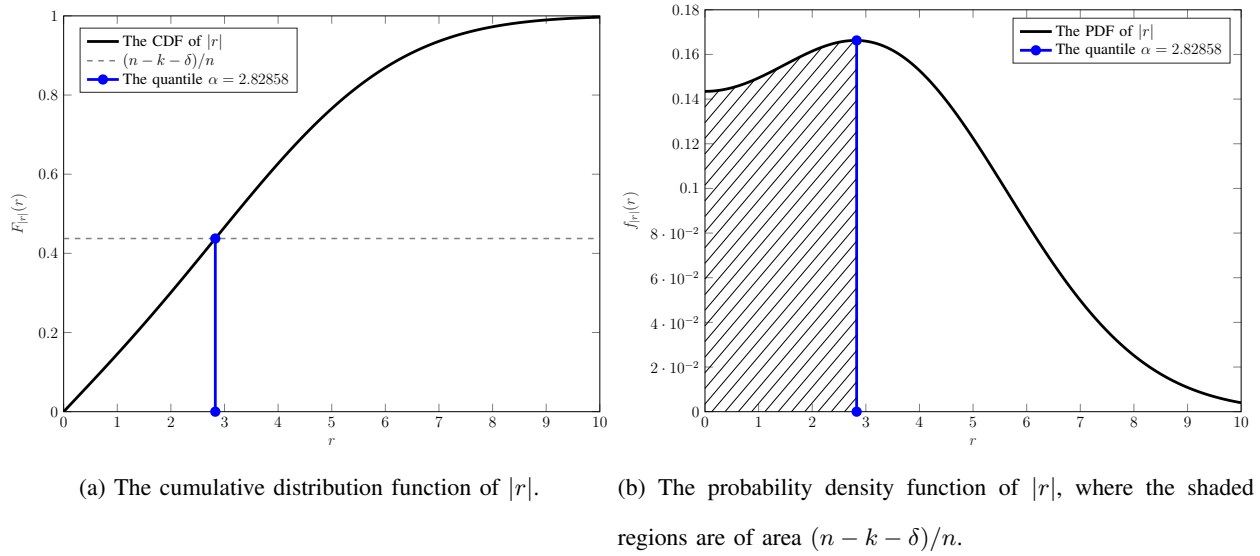


Fig. 2. Sketches of $(n - k - \delta)$ th n -quantile for $|r|$, where $(n, k, \delta) = (128, 64, 8)$ and $E_b/N_0 = 2.0$ dB.

Example 2. For better understanding of α , we have provided an example shown in Fig. 2 with a $\mathcal{C}[128, 64]$ code and $\delta = 8$.

Example 3. To illustrate the accuracy of the approximation, we compare τ_{SAI} with the statistic value of $\mathbb{E}[\Gamma(\tilde{e}_L)]$ by 10^6 simulations in Table I.

TABLE I
COMPARISON BETWEEN $\mathbb{E}[\Gamma(\tilde{e}_L)]$ AND τ_{SAI} , WHERE $(n, k, \delta) = (128, 64, 8)$.

| E_b/N_0 | $\mathbb{E}[\Gamma(\tilde{e}_L)]$ | $\tau_{\text{SAI}} (42)$ | Relative error |
|-----------|-----------------------------------|--------------------------|----------------|
| 0.0 dB | 12.038 | 12.079 | +0.346% |
| 0.5 dB | 12.222 | 12.276 | +0.437% |
| 1.0 dB | 12.282 | 12.341 | +0.483% |
| 1.5 dB | 12.181 | 12.246 | +0.535% |
| 2.0 dB | 11.895 | 11.963 | +0.575% |
| 2.5 dB | 11.413 | 11.477 | +0.563% |
| 3.0 dB | 10.727 | 10.787 | +0.557% |
| 3.5 dB | 9.860 | 9.910 | +0.500% |

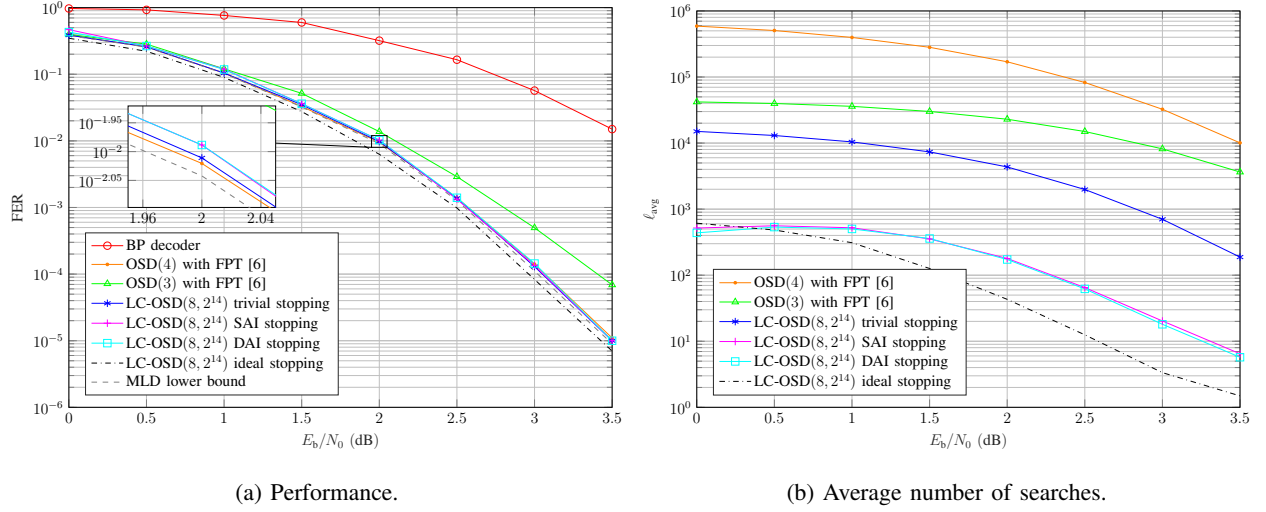


Fig. 3. Simulation results of the LDPC code $\mathcal{C}_1[128, 64]$ constructed in [17].

Example 4. Consider a rate-1/2 LDPC code $\mathcal{C}_1[128, 64]$ constructed in [17]. The constraint degree $\delta = 8$ and the maximum number of searches $\ell_{\max} = 2^{14}$. As shown in Fig. 3a, several decoders are implemented to demonstrate their performance, the frame error rate (FER), where the OSD with FPT [6] using trivial stopping criterion is a low-complexity implementation for OSD that has the same performance as OSD. The MLD lower bound in Fig. 3a is a simulated bound as presented in [18] and calculated by utilizing the LC-OSD (see Algorithm 3). Fig. 3b shows the average number of searches per frame for these decoders.

Algorithm 3 Lower bound on error counter for MLD**Input:** \mathbf{H} , δ , ℓ_{\max} , transmission-receiving pair (\mathbf{c}, \mathbf{y}) **Output:** MLD error is detected or not.

```

1: function MLD-ERROR-COUNTER( $\mathbf{H}, \delta, \ell_{\max}, \mathbf{c}, \mathbf{y}$ )
2:    $\hat{\mathbf{c}} \leftarrow \text{LC-OSD}(\mathbf{H}, \mathbf{y}, \delta, \ell_{\max}).$   $\triangleright$  Search for the optimal codeword by the LC-OSD.
3:   if  $f_{\mathbf{y}|\mathbf{c}}(\mathbf{y} | \hat{\mathbf{c}}) > f_{\mathbf{y}|\mathbf{c}}(\mathbf{y} | \mathbf{c})$  then  $\triangleright$  The optimal codeword  $\hat{\mathbf{c}}$  is more likely than the sent.
4:     return 1.  $\triangleright$  An MLD error occurs.
5:   else
6:     return 0.  $\triangleright$  No MLD error is detected.

```

For the reason that the LC-OSD with DAI stopping criterion has much lower time complexity without incurring obvious performance loss,⁷ we use the LC-OSD with DAI stopping criterion in the rest of this paper.

B. Influences of Decoding Parameters

Recalling that $\delta = 0$ implies the original OSD algorithm and $\delta = n - k$ implies the MLD, we can safely infer that, as the increase of δ , the maximum number of searches can be reduced. Therefore, in this subsection, we discuss the relationship between FER and the decoding parameters (δ, ℓ_{\max}) .

1) *The Channel of MRBs:* The channel of each MRB is nearly⁸ an “order statistic” AWGN channel

$$\tilde{y}_i = \tilde{x}_i + \tilde{w}_i, \quad n - k - \delta < i \leq n. \quad (44)$$

⁷This was verified by simulations in [7] and holds for a wide class of short codes.

⁸This is because the linear dependency of columns of \mathbf{H} and hence $|\tilde{\mathbf{r}}|$ is not always non-decreasing.

Formally, $|\tilde{y}_i|$ ($n - k - \delta < i \leq n$) is the i th order statistic of $|y_i|$ with probability density function given by

$$f_{|\tilde{y}_i|}(y) = \frac{n!}{(i-1)!(n-i)!} f_{|y|}(y) (F_{|y|}(y))^{i-1} (1 - F_{|y|}(y))^{n-i}, \quad (45)$$

where $f_{|y|}(\cdot)$ and $F_{|y|}(\cdot)$ are the probability density function and cumulative distribution function of $|y|$, respectively, i.e.,

$$f_{|y|}(y) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(y-1)^2}{2\sigma^2}\right) + \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(y+1)^2}{2\sigma^2}\right), \quad y \geq 0, \quad (46)$$

$$F_{|y|}(y) = \int_0^y f_{|y|}(t) dt = \frac{1}{2} \operatorname{erf}\left(\frac{y-1}{\sqrt{2}\sigma}\right) + \frac{1}{2} \operatorname{erf}\left(\frac{y+1}{\sqrt{2}\sigma}\right), \quad y \geq 0, \quad (47)$$

and $\operatorname{erf}(\cdot)$ is the Gaussian error function defined as

$$\operatorname{erf}(x) \triangleq \frac{2}{\sqrt{\pi}} \int_0^x \exp(-t^2) dt. \quad (48)$$

2) *The MRB Code:* After preprocessing, the locally constrained matrix \mathbf{P}_2 , defining the parity-check matrix of the MRB code $\mathcal{C}_{\text{MRB}}[k + \delta, k]$ of MRBs, can be approximately regarded as a random binary matrix.⁹ Denote the rank (in the order of likelihood) of the correct TEP of MRBs by L , the MRBs' FER ε_{MRB} of the optimal list decoder¹⁰ with the list size of ℓ_{\max} is defined as

$$\varepsilon_{\text{MRB}} \triangleq \mathbb{P}[L > \ell_{\max}], \quad (49)$$

which can be approximated by the FER of a random code¹¹ $\widehat{\mathcal{C}}_{\text{MRB}}[k + \delta, k]$, i.e.,

$$\varepsilon_{\text{MRB}} \approx \widehat{\varepsilon}_{\text{MRB}} \triangleq \mathbb{P}[\widehat{L} > \ell_{\max}] \quad (50)$$

where \widehat{L} denotes the rank of the correct codeword of $\widehat{\mathcal{C}}_{\text{MRB}}$.

⁹The randomness lies in the fact that the receiving vector \mathbf{y} and hence the choices of MRBs are random.

¹⁰Here, the optimal list decoder generates a list containing the ℓ_{\max} lightest TEPs. In other words, ε_{MRB} stands for the probability that the true $\tilde{\mathbf{e}}_{\text{R}}$ (or equivalently, the correct codeword) has a rank (in terms of soft weight) greater than ℓ_{\max} .

¹¹With a slight abuse of notation, $\widehat{\mathcal{C}}_{\text{MRB}}[k + \delta, k]$ is used to denote a code of length $k + \delta$ and size 2^k , whose codewords are chosen independently and uniformly at random from $\mathbb{F}_2^{k+\delta}$.

In the following, instead of \tilde{e}_R , \tilde{y}_R and \tilde{z}_R , we simply use the notations \tilde{e} , \tilde{y} and \tilde{z} to denote the MRB part of the TEP, received vector, and bit-wise hard decision vector, respectively. We introduce a set

$$D \triangleq \{\tilde{\mathbf{f}} \in \mathbb{F}_2^{k+\delta} : \Gamma(\tilde{\mathbf{f}}) \leq \Gamma(\tilde{\mathbf{e}})\} \quad (51)$$

of all sequences not heavier than \tilde{e} . By definition, we have

$$\tilde{e} = \arg \max_{\tilde{\mathbf{f}} \in D} \Gamma(\tilde{\mathbf{f}}). \quad (52)$$

From the definition of \hat{L} , we have

$$\hat{L} = |D \cap (\hat{\mathbf{z}} - \mathcal{C}_{\text{MRB}})| = \sum_{\tilde{\mathbf{f}} \in D} \mathbb{I}[\tilde{\mathbf{f}} \in \hat{\mathbf{z}} - \mathcal{C}_{\text{MRB}}] \quad (53)$$

where $\mathbb{I}[\cdot]$ is the indicator function. Let

$$\hat{L}' \triangleq \hat{L} - 1 \quad (54)$$

be the number of TEPs which are strictly lighter¹² than \tilde{e} , i.e.,

$$\hat{L}' = \sum_{\tilde{\mathbf{f}} \in D \setminus \{\tilde{\mathbf{e}}\}} \mathbb{I}[\tilde{\mathbf{f}} \in \hat{\mathbf{z}} - \mathcal{C}_{\text{MRB}}]. \quad (55)$$

Therefore, whenever D is given, \hat{L}' is randomly distributed as the binomial distribution

$$\hat{L}' \sim \text{Binomial}\left(|D| - 1, \frac{2^k - 1}{2^{k+\delta} - 1}\right) \quad (56)$$

with the probability mass function (PMF) given by

$$\mathbb{P}[\hat{L}' = \ell' \mid D] = \binom{|D| - 1}{\ell'} \left(\frac{2^k - 1}{2^{k+\delta} - 1}\right)^{\ell'} \left(1 - \frac{2^k - 1}{2^{k+\delta} - 1}\right)^{|D| - 1 - \ell'} \quad (57)$$

Hence,

$$\hat{\varepsilon}_{\text{MRB}} = \mathbb{P}[\hat{L} > \ell_{\max}] = \mathbb{P}[\hat{L}' \geq \ell_{\max}] = \mathbb{E}_D[\mathbb{P}[\hat{L}' \geq \ell_{\max} \mid D]]. \quad (58)$$

¹²Since the probability measure is zero that any two TEPs have equal soft weights (continuous random variables), we can safely assume that any two TEPs have distinct soft weights without introducing approximation.

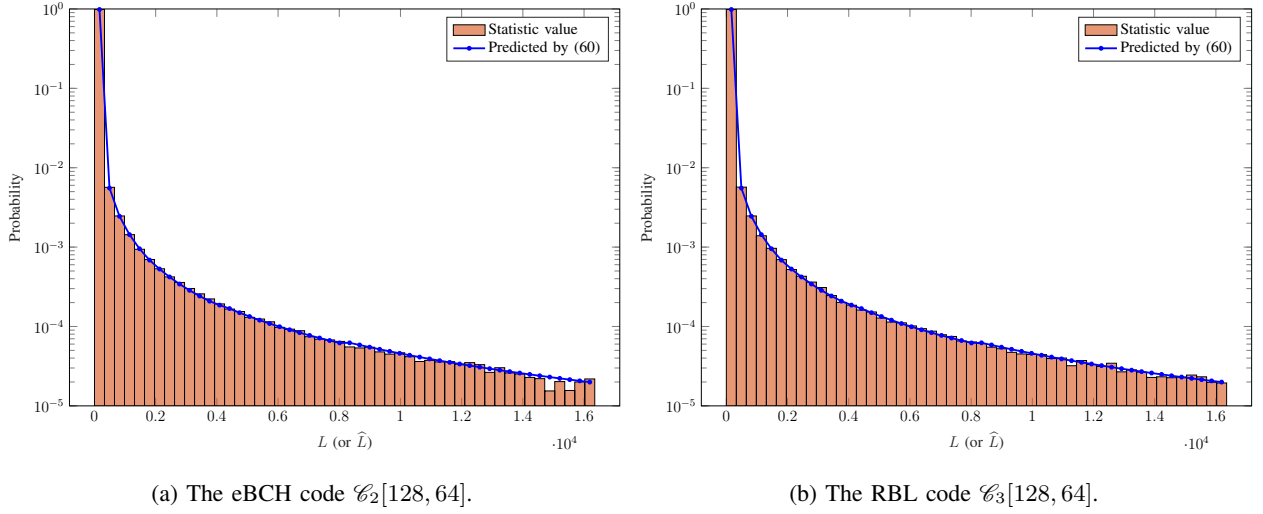


Fig. 4. The statistical histogram of L , the rank of the correct codeword, in the case of $\delta = 8$ and $E_b/N_0 = 2.0$ dB.

It is worth mentioning that the definition (50) of $\hat{\varepsilon}_{\text{MRB}}$ does not depend on the structure of any specific code. Also notice that, ε_{MRB} and $\hat{\varepsilon}_{\text{MRB}}$ are functions of ℓ_{\max} , denoting as $\varepsilon_{\text{MRB}}(\ell_{\max})$ and $\hat{\varepsilon}_{\text{MRB}}(\ell_{\max})$ respectively. Then we can write the PMF of L (and \hat{L}) by

$$\mathbb{P}[L = \ell] = \mathbb{P}[L > \ell - 1] - \mathbb{P}[L > \ell] = \varepsilon_{\text{MRB}}(\ell - 1) - \varepsilon_{\text{MRB}}(\ell), \quad (59)$$

$$\mathbb{P}[\hat{L} = \ell] = \mathbb{P}[\hat{L} > \ell - 1] - \mathbb{P}[\hat{L} > \ell] = \hat{\varepsilon}_{\text{MRB}}(\ell - 1) - \hat{\varepsilon}_{\text{MRB}}(\ell). \quad (60)$$

Example 5. To illustrate the accuracy of the approximation, we use $\mathbb{P}[L = \ell] \approx \mathbb{P}[\hat{L} = \ell]$ to estimate the PMF of L and compare it with the statistic value by simulating two different codes, an extended Bose-Chaudhuri-Hocquenghem (eBCH) code $\mathcal{C}_2[128, 64]$ (Fig. 4a) and a random binary linear (RBL) code $\mathcal{C}_3[128, 64]$ whose parity-check matrix is randomly constructed (Fig. 4b). As seen from Fig. 4, the statistic values are quite close to the predicted values from (50) and (60) and are less relevant to the structure of the code. Hence, we have $\mathbb{E}[L] \approx \mathbb{E}[\hat{L}]$ and use $\hat{\varepsilon}_{\text{MRB}}$ to estimate ε_{MRB} in the following analysis.

Conditional on the transmitted codeword being in the list, we have

$$\mathbb{E}[L \mid L \leq \ell_{\max}] \approx \mathbb{E}[\hat{L} \mid \hat{L} \leq \ell_{\max}], \quad (61)$$

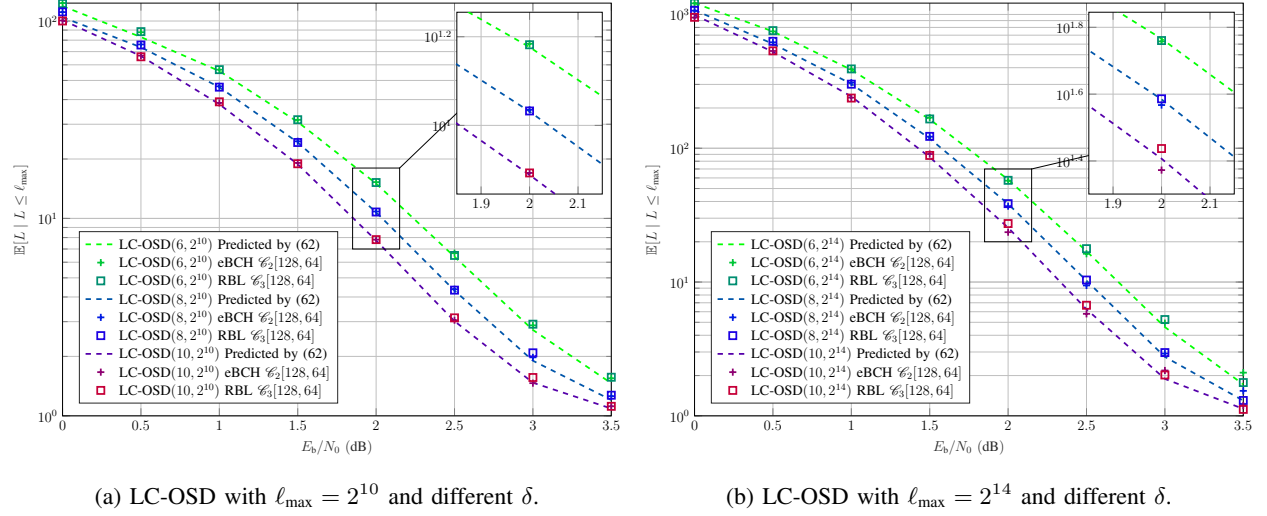


Fig. 5. Conditional expectation of L of the eBCH code $\mathcal{C}_2[128, 64]$ and the RBL code $\mathcal{C}_3[128, 64]$ along with their predictions.

where $\mathbb{E}[\hat{L} \mid \hat{L} \leq \ell_{\max}]$ can be calculated (see Appendix C for details) as

$$\mathbb{E}[\hat{L} \mid \hat{L} \leq \ell_{\max}] = \frac{1}{1 - \hat{\varepsilon}_{\text{MRB}}(\ell_{\max})} \left(\sum_{\ell=0}^{\ell_{\max}-1} \hat{\varepsilon}_{\text{MRB}}(\ell) - \ell_{\max} \cdot \hat{\varepsilon}_{\text{MRB}}(\ell_{\max}) \right). \quad (62)$$

Example 6. We have simulated the left-hand side (LHS) of (61) of the eBCH code $\mathcal{C}_2[128, 64]$ as well as the RBL code $\mathcal{C}_3[128, 64]$. The results are shown in Fig. 5. Also shown in Fig. 5 are the numerical values calculated by (62). We can see that the simulated values are close to each other and also to the calculated values, validating the assumption on the randomness of \mathcal{C}_{MRB} .

3) *Performance Bound:* Denote by ε and ε_{ML} the FER of the LC-OSD and MLD, respectively. Then we have

$$\varepsilon = \mathbb{P}[\{\tilde{e} \text{ is not in the list}\}] + \mathbb{P}[\{\tilde{e} \text{ is in the list but is not the most likely one}\}] \quad (63)$$

$$\leq \mathbb{P}[\{\tilde{e} \text{ is not in the list}\}] + \mathbb{P}[\{\tilde{e} \text{ is not the most likely one}\}] \quad (64)$$

$$= \varepsilon_{\text{MRB}} + \varepsilon_{\text{ML}} \quad (65)$$

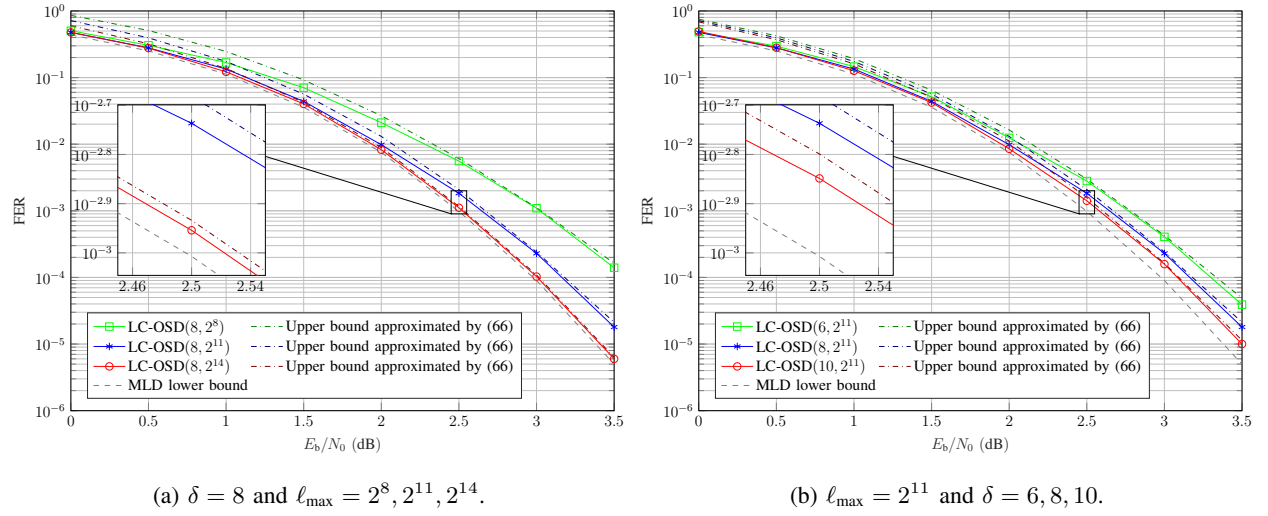


Fig. 6. Performance and approximate upper bound of the RBL code $\mathcal{C}_3[128, 64]$.

Hence, the performance gap between the LC-OSD and the MLD can be approximately upper bounded by $\widehat{\varepsilon}_{\text{MRB}}$ as,

$$\varepsilon - \varepsilon_{\text{ML}} \leq \varepsilon_{\text{MRB}} \approx \widehat{\varepsilon}_{\text{MRB}} \quad (66)$$

which does not depend on the structure of the code.

Example 7. Fig. 6 shows the performance of the RBL code $\mathcal{C}_3[128, 64]$. We also plotted the approximate upper bound (66) of $\mathcal{C}_3[128, 64]$. As shown in Fig. 6 we can observe that the performance of $\mathcal{C}_3[128, 64]$ improves as ℓ_{\max} or δ increases. Additionally, we note that the approximate upper bounds are highly accurate due to the accuracy of (50), which has already been verified in Example 5 and Example 6.

C. Decoding Parameters Tuning

The tight performance upper bound deduced in Sec. III-B3 for the LC-OSD becomes a helpful tool in many situations. In this subsection, we will give examples that explain how we could tune the decoding parameters (δ, ℓ_{\max}) to satisfy the diversified requirements depending on application scenarios.

To evaluate the time consumption of the LC-OSD, we need to write out the time factor in (26) and (27), namely,

$$T_{\text{avg}} \approx \rho_1 \cdot n(n-k)(n-k-\delta) + \rho_2 \cdot 2^\delta(k+\delta) + \rho_3 \cdot \ell_{\text{avg}}(n-k-\delta)(k+\delta), \quad (67)$$

$$T_{\text{max}} \approx \rho_1 \cdot n(n-k)(n-k-\delta) + \rho_2 \cdot 2^\delta(k+\delta) + \rho_3 \cdot \ell_{\text{max}}(n-k-\delta)(k+\delta), \quad (68)$$

where ρ_i ($i = 1, 2, 3$) are called the time factor constants. We can measure these time factor constants by implementing the LC-OSD on a specific computer (or device).

Example 8. Table II shows the values of these time factor constants for reference.¹³

TABLE II
REFERENCE VALUES OF TIME FACTORS.

| Time factor constants | ρ_1 | ρ_2 | ρ_3 |
|-----------------------|----------|----------|----------|
| Measured values (ns) | 0.0816 | 26.4 | 0.728 |

Then we may tune the decoding parameters as required by applications.

1) Minimum Average Time Complexity: In the scenarios where the power consumption is concerned, we need to optimize $(\delta, \ell_{\text{max}})$ to minimize the average time complexity T_{avg} .

Example 9. We have simulated the average number of searches of the LC-OSD for the eBCH code $\mathcal{C}_2[128, 64]$ using different decoding parameters $(\delta, \ell_{\text{max}})$. From Fig. 7a, we see that, the worse the channel is, the higher complexity of the LC-OSD is, as in line with our intuition. We have implemented the LC-OSD algorithm with SLVA on the personal desktop and tried to optimize the speed as much as possible.

¹³We have measured these values on a personal desktop with a CPU of 11th Gen Intel(R) Core(TM) i7-11700F @ 2.50GHz.

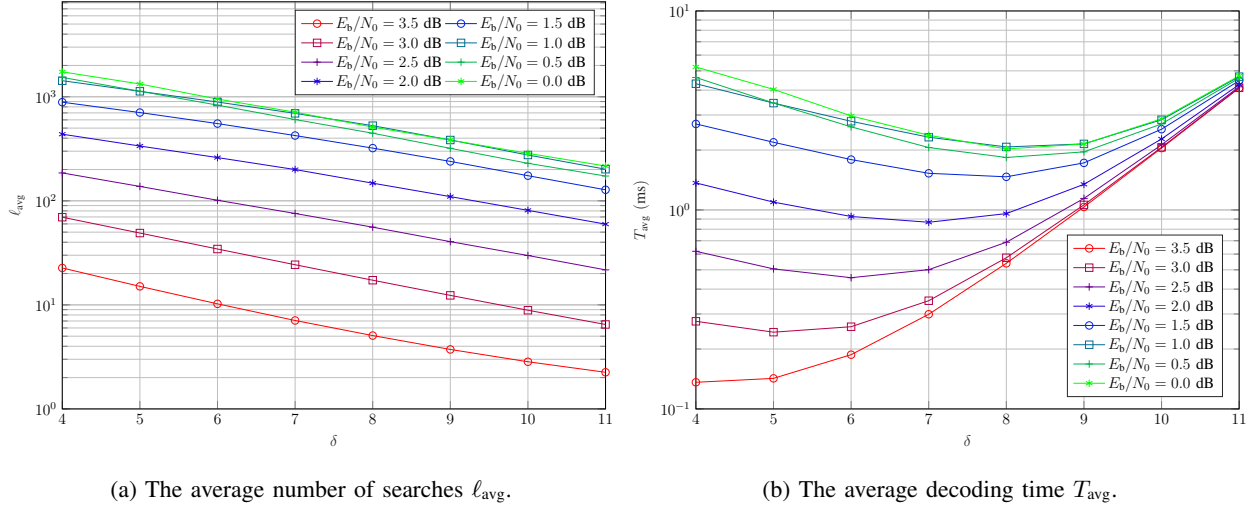


Fig. 7. Average time complexity of the LC-OSD with $\ell_{\text{max}} = 2^{14}$ for the eBCH code $\mathcal{C}_2[128, 64]$.

2) *Minimum Worst-case Time Complexity:* In the scenarios where the maximum delay is concerned, we need to limit the maximum list size ℓ_{max} to control the worst-case decoding time.

Recalling the definition in (50), $\widehat{\varepsilon}_{\text{MRB}}$ is a function of $(\delta, \ell_{\text{max}})$, denoted as $\widehat{\varepsilon}_{\text{MRB}}(\delta, \ell_{\text{max}})$. Intuitively, $\widehat{\varepsilon}_{\text{MRB}}(\delta, \ell_{\text{max}})$ is a non-increasing function of ℓ_{max} for a fixed δ . Therefore, given a target FER $\widehat{\varepsilon}_{\text{MRB}}^*$, let $\ell_{\text{max}}^*(\delta)$ be the minimum ℓ_{max} for a fixed δ under the constraint of the target FER, i.e.,

$$\ell_{\text{max}}^*(\delta) \triangleq \min_{\widehat{\varepsilon}_{\text{MRB}}(\delta, \ell_{\text{max}}) \leq \widehat{\varepsilon}_{\text{MRB}}^*} \ell_{\text{max}}. \quad (69)$$

To meet the requirement of minimum worst-case time complexity, we can adjust the decoding parameter to $(\ell_{\text{max}}^*, \delta)$.

Example 10. Intuitively, as δ increases, the constraint specified by \mathbf{P}_2 becomes more “global”, and hence the number of searches required for the LC-OSD to have near MLD performance decreases. In the extreme case when δ reaches its maximum value $(n - k)$, $\ell_{\text{max}} = 1$ is sufficient. This intuition can be verified by the simulation. Fig. 8a shows the required ℓ_{max} versus δ for the LC-OSD to have near MLD performance. As expected, the larger δ is, the smaller ℓ_{max} is

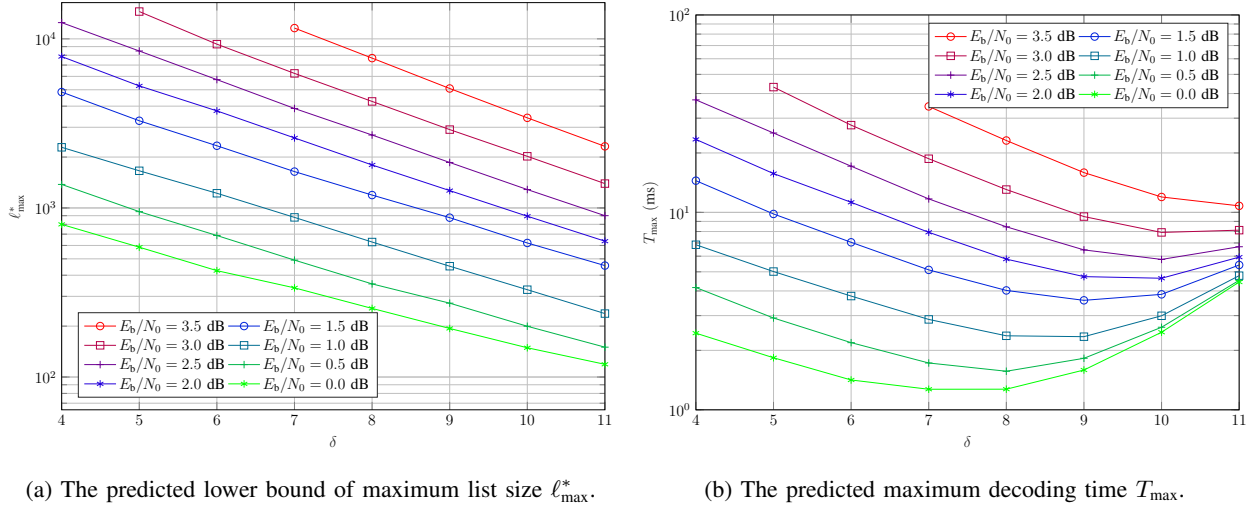


Fig. 8. Tuning parameters on (69) for the LC-OSD of the eBCH code $\mathcal{C}_2[128, 64]$ to have near MLD performance ($\hat{\varepsilon}_{\text{MRB}}^* = \varepsilon_{\text{ML}}$).

required.

Example 11. At first glance, it seems that a larger δ has higher decoding efficiency. However, this is not always the case. On the one hand, smaller δ values lead to larger ℓ_{\max} values, resulting in slower searches. On the other hand, large δ values (which result in small ℓ_{\max}) can have significant delays caused by initialization (as discussed in Sec. II-C1). As a result, the maximum time complexity is dominated by search when δ is small and by initialization when δ is large. Therefore, it is meaningful to carefully tune the values of δ and ℓ_{\max} for achieving high decoding efficiency. This has been verified by the simulation shown in Fig. 8b.

3) Limited Maximum Space Complexity: In the scenarios where the maximum memory usage is concerned, we need to consider the problem of parameter tuning in the case of limited space. The examples in these cases are similar to the previous scenarios and will not be repeated here.

IV. FURTHER COMPARISONS

We simulate in this section several well-known codes to show the universality, efficiency and flexibility of the LC-OSD. We also present simulation results of some other decoding algorithms

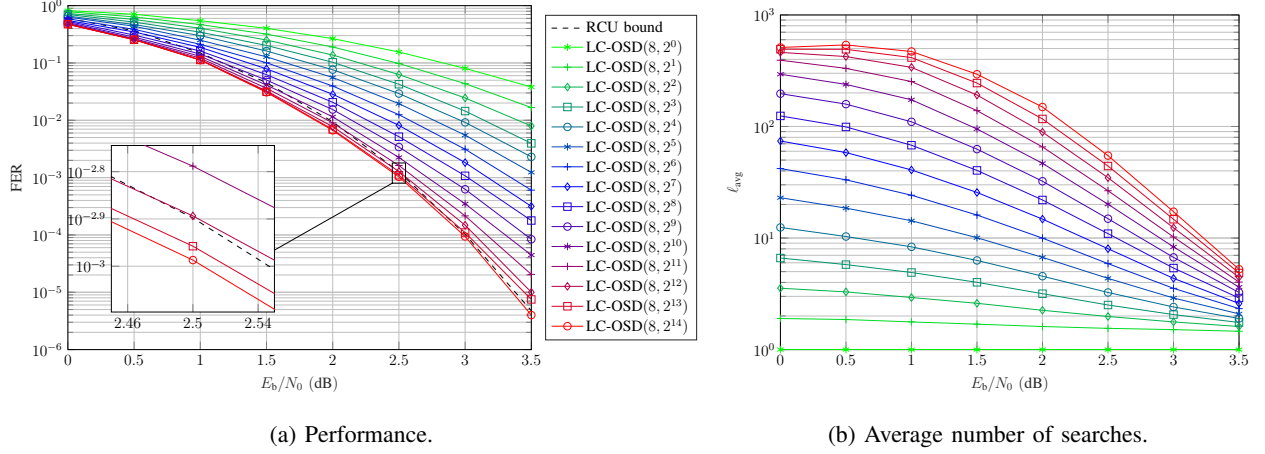
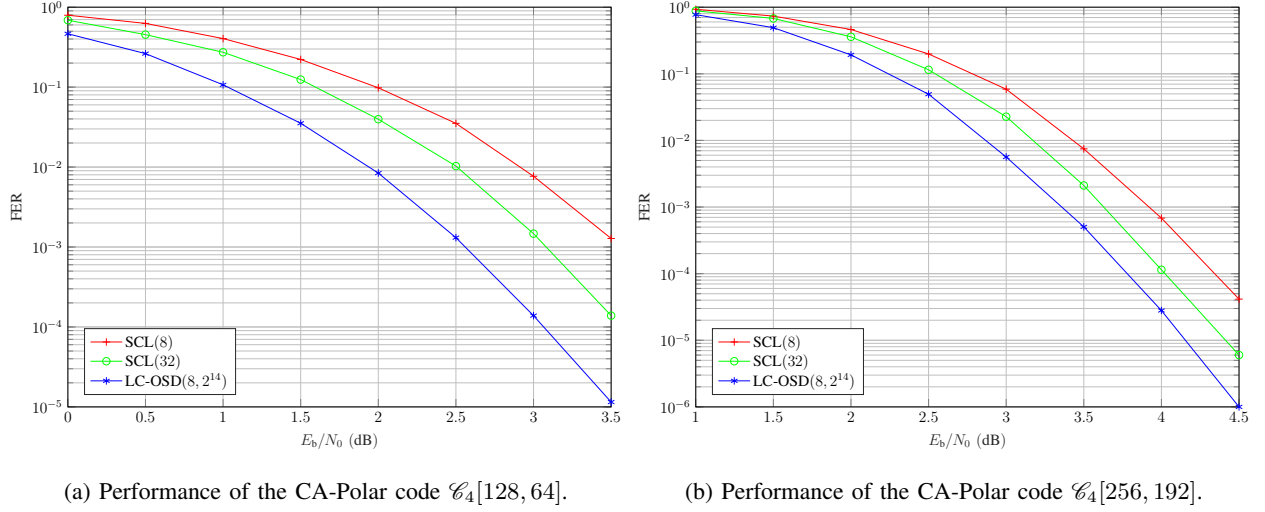
Fig. 9. Simulation results of the eBCH code $\mathcal{C}_2[128, 64]$.

Fig. 10. Simulation results of the CA-Polar codes, comparing the LC-OSD with the SCL decoding [19].

for comparison.

Example 12. Fig. 9 shows that the performance of the eBCH code $\mathcal{C}_2[128, 64]$ improves as ℓ_{\max} increases but saturates at a certain level of ℓ_{\max} . This is reasonable because the performance of the LC-OSD is lower bounded by that of the MLD, regardless of how large ℓ_{\max} is.

Example 13. We present the performance of two CRC-aided polar (CA-polar) codes with

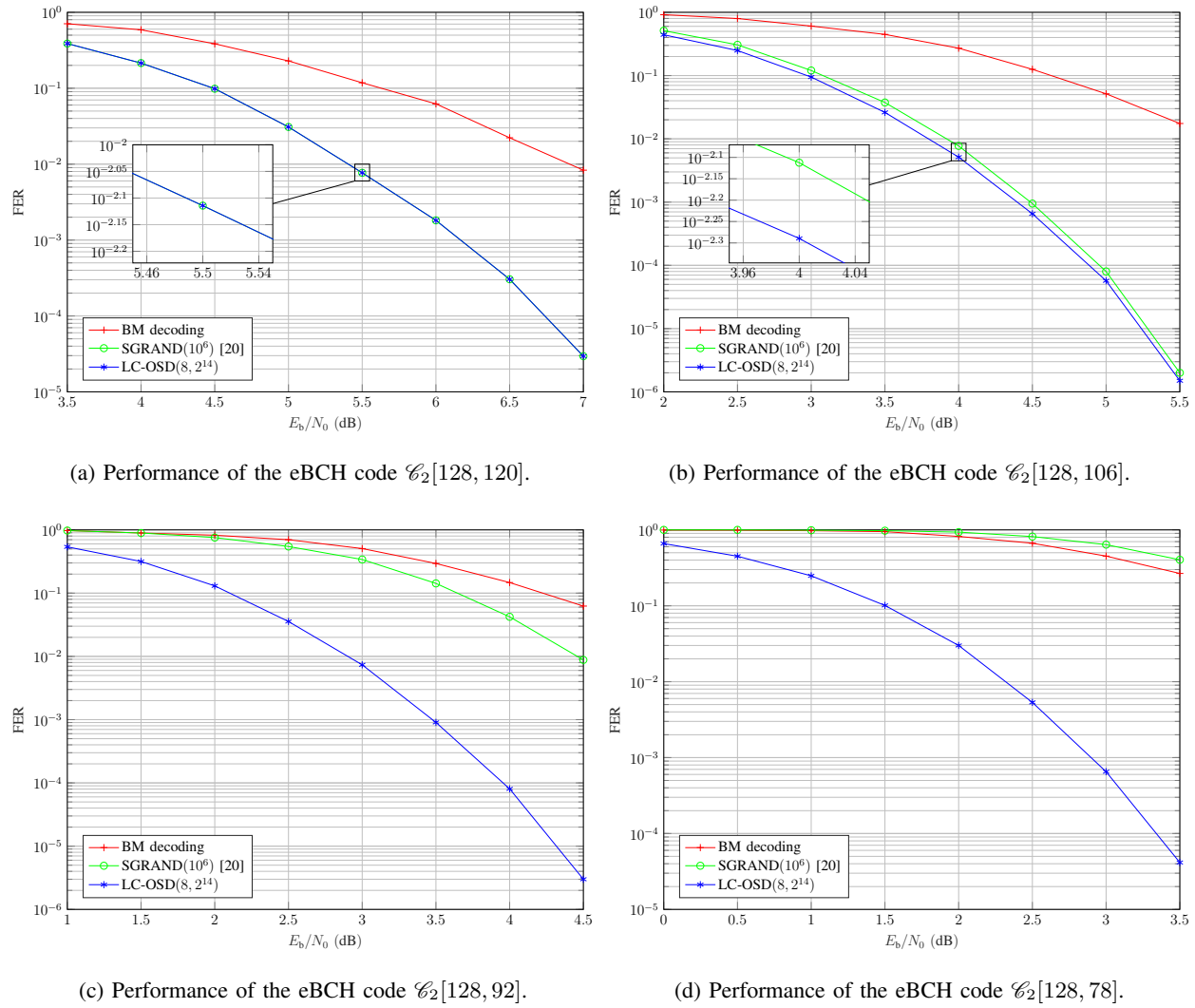


Fig. 11. Simulation results of the eBCH codes, comparing the LC-OSD with BM decoding and SGRAND [20].

different lengths and the same redundancy, namely $\mathcal{C}_4[128, 64]$ and $\mathcal{C}_4[256, 192]$, both of which are with 11 bits CRCs. Fig. 10 shows that the LC-OSD decoding outperforms the successive cancellation list (SCL) decoding [19].

Example 14. We compare the LC-OSD with the soft GRAND (SGRAND) [20], [21]. Unlike OSD, the SGRAND is a decoding algorithm that does not use Gaussian elimination. We have compared the performance of the LC-OSD, SGRAND [20], and BM algorithm for length-128

eBCH codes with different rates. Simulation results show that the LC-OSD with appropriate parameters performs better than SGRAND for the tested eBCH codes.

V. CONCLUSION

In this paper, we have presented an extended version of OSD called the locally constrained OSD (LC-OSD) algorithm. The LC-OSD uses more MRBs than the original OSD and solves the local constraint by the SLVA. We have also presented several early stopping criteria for the LC-OSD that significantly reduce computational complexity without noticeable performance loss. To analyze the performance of LC-OSD, we have proposed a method that uses random code to approximate the local constraint over MRBs. Several examples show that the approximation is quite accurate and does not depend on the structure of the codes. Based on the high accuracy of the approximation, we have proposed a tight approximate upper bound of the LC-OSD, which is applicable to different codes and decoding parameters. The random coding approach can lead to many researches and applications related to the LC-OSD, one of which is tuning the decoding parameters according to applications.

ACKNOWLEDGMENT

The authors would like to express their gratitude to Dr. Suihua Cai for his enlightening discussions and to Dr. Marc Fossorier for bringing [8] to their attention, of which they were previously unaware.

APPENDIX A

CALCULATION OF THE DAI THRESHOLD τ_{DAI}

Here, we show how to calculate the DAI threshold τ_{DAI} (41). Upon receiving \mathbf{y} , the LLR vector \mathbf{r} can be calculated out by (6). Then, by definition, we have

$$\Gamma(\tilde{\mathbf{e}}_{\text{L}}) = \langle \tilde{\mathbf{e}}_{\text{L}}, |\tilde{\mathbf{r}}_{\text{L}}| \rangle = \sum_{i=1}^{n-k-\delta} \tilde{e}_i |\tilde{r}_i|, \quad (70)$$

and hence

$$\tau_{\text{DAI}} = \mathbb{E}[\Gamma(\tilde{\mathbf{e}}_{\text{L}}) \mid \mathbf{r}] \quad (71)$$

$$= \mathbb{E} \left[\sum_{i=1}^{n-k-\delta} \tilde{e}_i |\tilde{r}_i| \mid \mathbf{r} \right] \quad (72)$$

$$= \mathbb{E} \left[\sum_{i=1}^{n-k-\delta} \tilde{e}_i |\tilde{r}_i| \mid \tilde{\mathbf{r}} \right] \quad (73)$$

$$= \sum_{i=1}^{n-k-\delta} \mathbb{E}[\tilde{e}_i |\tilde{r}_i| \mid \tilde{\mathbf{r}}] \quad (74)$$

$$= \sum_{i=1}^{n-k-\delta} \mathbb{E}[\tilde{e}_i |\tilde{r}_i| \mid \tilde{r}_i] \quad (75)$$

$$= \sum_{i=1}^{n-k-\delta} \mathbb{E}[\tilde{e}_i \mid \tilde{r}_i] \cdot |\tilde{r}_i| \quad (76)$$

The i th bit of true TEP \tilde{e}_i is a random variable with

$$\mathbb{P}[\tilde{e}_i = 0 \mid \tilde{r}_i] = \frac{1}{1 + \exp(-|\tilde{r}_i|)}, \quad (77)$$

$$\mathbb{P}[\tilde{e}_i = 1 \mid \tilde{r}_i] = \frac{1}{1 + \exp(|\tilde{r}_i|)}. \quad (78)$$

Therefore,

$$\mathbb{E}[\tilde{e}_i \mid \tilde{r}_i] = \mathbb{P}[\tilde{e}_i = 1 \mid \tilde{r}_i] = \frac{1}{1 + \exp(|\tilde{r}_i|)} \quad (79)$$

and

$$\tau_{\text{DAI}} = \sum_{i=1}^{n-k-\delta} \frac{1}{1 + \exp(|\tilde{r}_i|)} \cdot |\tilde{r}_i|. \quad (80)$$

APPENDIX B

CALCULATION OF THE SAI THRESHOLD τ_{SAI}

Here, we show how to calculate the SAI threshold τ_{SAI} (43). The $(n - k - \delta)$ th n -quantile for the random variable $|r|$ can be defined as

$$\alpha \triangleq F_{|r|}^{-1} \left(\frac{n - k - \delta}{n} \right), \quad (81)$$

where $F_{|r|}^{-1}(\cdot)$ is the inverse function of $F_{|r|}(\cdot)$ which is the cumulative distribution function of $|r|$, i.e.,

$$f_{|r|}(r) = \frac{\sigma}{2\sqrt{2\pi}} \exp \left(-\frac{(\sigma^2 r - 2)^2}{8\sigma^2} \right) + \frac{\sigma}{2\sqrt{2\pi}} \exp \left(-\frac{(\sigma^2 r + 2)^2}{8\sigma^2} \right), \quad r \geq 0, \quad (82)$$

$$F_{|r|}(r) = \int_0^r f_{|r|}(t) dt = \frac{1}{2} \operatorname{erf} \left(\frac{\sigma^2 r - 2}{2\sqrt{2}\sigma} \right) + \frac{1}{2} \operatorname{erf} \left(\frac{\sigma^2 r + 2}{2\sqrt{2}\sigma} \right), \quad r \geq 0. \quad (83)$$

As given in (79), the conditional expectation of $e|r|$ is

$$\gamma(r) \triangleq \mathbb{E}[e|r| \mid r] \quad (84)$$

$$= |r| \cdot \mathbb{E}[e \mid r] \quad (85)$$

$$= \frac{|r|}{1 + \exp(|r|)} \quad (86)$$

Therefore, (43) can be deduced by

$$\tau_{\text{SAI}} = (n - k - \delta) \mathbb{E}[e|r| \mid |r| \leq \alpha] \quad (87)$$

$$= (n - k - \delta) \mathbb{E}[\gamma(r) \mid |r| \leq \alpha] \quad (88)$$

$$= \frac{n - k - \delta}{\mathbb{P}[|r| \leq \alpha]} \int_0^\alpha \gamma(r) f_{|r|}(r) dr \quad (89)$$

$$= \frac{n - k - \delta}{F_{|r|}(\alpha)} \int_0^\alpha \gamma(r) f_{|r|}(r) dr \quad (90)$$

$$= n \int_0^\alpha \gamma(r) f_{|r|}(r) dr \quad (91)$$

$$= n \int_0^\alpha \frac{|r|}{1 + \exp(|r|)} f_{|r|}(r) dr \quad (92)$$

APPENDIX C

CALCULATION OF THE CONDITIONAL EXPECTATION $\mathbb{E}[\hat{L} \mid \hat{L} \leq \ell_{\max}]$

The conditional expectation (62) can be deduced by

$$\mathbb{E}[\hat{L} \mid \hat{L} \leq \ell_{\max}] = \sum_{\ell=1}^{\infty} \ell \cdot \mathbb{P}[\hat{L} = \ell \mid \hat{L} \leq \ell_{\max}] \quad (93)$$

$$= \frac{1}{\mathbb{P}[\hat{L} \leq \ell_{\max}]} \sum_{\ell=1}^{\ell_{\max}} \ell \cdot \mathbb{P}[\hat{L} = \ell] \quad (94)$$

$$= \frac{1}{\mathbb{P}[\hat{L} \leq \ell_{\max}]} \sum_{\ell=1}^{\ell_{\max}} \ell \cdot (\hat{\varepsilon}_{\text{MRB}}(\ell-1) - \hat{\varepsilon}_{\text{MRB}}(\ell)) \quad (95)$$

$$= \frac{1}{\mathbb{P}[\hat{L} \leq \ell_{\max}]} \left(\sum_{\ell=1}^{\ell_{\max}} \ell \cdot \hat{\varepsilon}_{\text{MRB}}(\ell-1) - \sum_{\ell=1}^{\ell_{\max}} \ell \cdot \hat{\varepsilon}_{\text{MRB}}(\ell) \right) \quad (96)$$

$$= \frac{1}{\mathbb{P}[\hat{L} \leq \ell_{\max}]} \left(\sum_{\ell=0}^{\ell_{\max}-1} (\ell+1) \cdot \hat{\varepsilon}_{\text{MRB}}(\ell) - \sum_{\ell=0}^{\ell_{\max}} \ell \cdot \hat{\varepsilon}_{\text{MRB}}(\ell) \right) \quad (97)$$

$$= \frac{1}{\mathbb{P}[\hat{L} \leq \ell_{\max}]} \left(\sum_{\ell=0}^{\ell_{\max}-1} \hat{\varepsilon}_{\text{MRB}}(\ell) - \ell_{\max} \cdot \hat{\varepsilon}_{\text{MRB}}(\ell_{\max}) \right) \quad (98)$$

$$= \frac{1}{1 - \hat{\varepsilon}_{\text{MRB}}(\ell_{\max})} \left(\sum_{\ell=0}^{\ell_{\max}-1} \hat{\varepsilon}_{\text{MRB}}(\ell) - \ell_{\max} \cdot \hat{\varepsilon}_{\text{MRB}}(\ell_{\max}) \right). \quad (99)$$

APPENDIX D

LIST VITERBI DECODING ALGORITHM

Here we present for completeness the list Viterbi decoding algorithms (LVAs) (including the parallel LVA, see, for example, [22], and the serial LVA), which produce an ordered list of the ℓ best candidates.

A. Trellis and Viterbi Algorithm

A code (block or convolutional) can be represented graphically by a trellis [23]–[25], which makes it possible (also convenient) to implement the MLD with reduced complexity. The most well-known trellis-based MLD algorithm is the Viterbi algorithm.

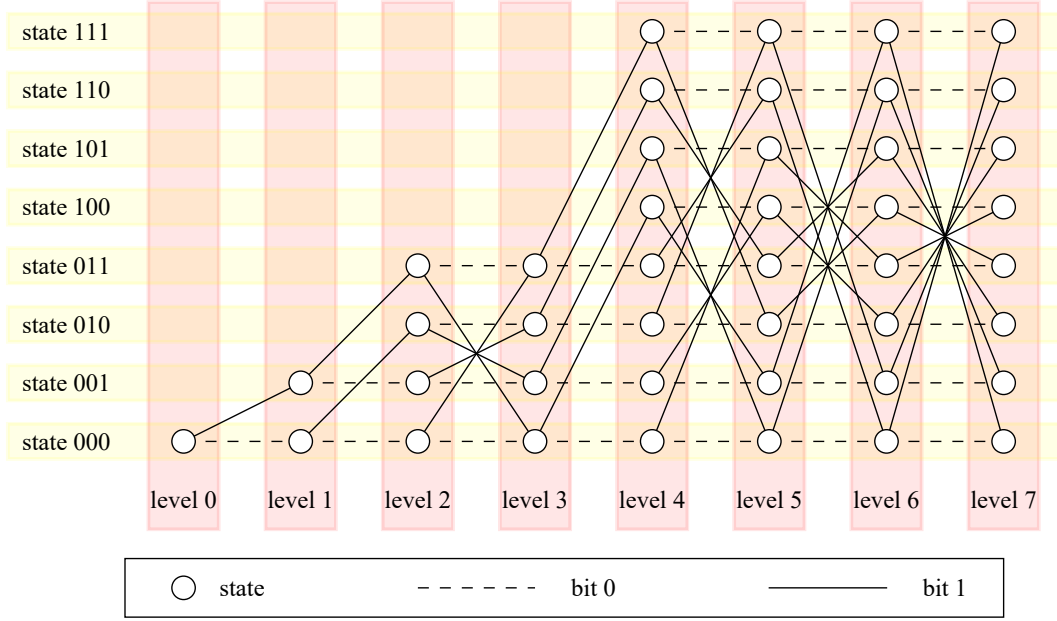


Fig. 12. The trellis representation of the $(7, 4)$ Hamming code for finding error patterns. The starting state is $\mathbf{0}$, while the ending state is $\mathbf{P}\tilde{\mathbf{z}}^T$.

A binary linear code $\mathcal{C}[N, K]$ with a parity-check matrix \mathbf{P} can be represented by a trellis with N sections and at most 2^{N-K} states at each section. The state space at level- i , written as $\Sigma_i(\mathcal{C})$, is the set of allowable states at the i th level. In the trellis, every vertex at level- i has a path from the initial state. Every vertex in the trellis has at least one incoming edge except for the initial state and at most two outgoing edges except for the states at the last level. For $i > 0$, the two outgoing edges from a state at the $(i - 1)$ -th level represent the bit 0 and bit 1, leading the state $\sigma \in \Sigma_{i-1}(\mathcal{C})$ to $(\sigma + \mathbf{0}) \in \Sigma_i(\mathcal{C})$ and $(\sigma + \mathbf{P}_i) \in \Sigma_i(\mathcal{C})$, respectively, where \mathbf{P}_i is the i th column of \mathbf{P} .

Example 15. Let us consider the $(7, 4)$ Hamming code with the parity check matrix

$$\mathbf{P} = \begin{bmatrix} 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{bmatrix} \quad (100)$$

whose trellis representation¹⁴ is shown in Fig. 12. It is worth noting that, Fig. 12 is not the trellis corresponding to the Hamming code but to its syndromes. There are $N + 1 = 8$ levels (indexed from 0 to 7) and at most $2^{N-K} = 8$ states (indexed from 000 to 111 in binary form) at each level. The white circles stand for states, and the dashed and solid edges for the bit 0 and 1, respectively.

Once a trellis of a code is constructed, the ML codeword could be obtained by running the Viterbi algorithm over the trellis. Equivalently, the Viterbi algorithm can be implemented to find the shortest path (lightest error pattern) e from the initial state $\mathbf{0}$ to the final state $\mathbf{P}\mathbf{z}^T$ over the trellis, where the soft weights of the dashed edges are 0 and that of the solid edges are the bit reliability $|r_i|$. See the discussion in Sec. II.

B. Parallel LVA

The parallel LVA (PLVA) searches for the ℓ_{\max} best paths at the same time by updating the ℓ_{\max} best paths into each state level by level.

C. Serial LVA

The serial LVA (SLVA) searches for the ℓ_{\max} most probable candidates, one by one, starting with the most probable path. The main advantage of this algorithm is that the ℓ th best path is computed only when the previously found $\ell - 1$ paths are not good enough, such as not fulfilling additional constraints. More importantly, after the first path is calculated, the SLVA can produce each following path in linear time and space complexity, which is done by using the intermediate

¹⁴The edges in Fig. 12 are directed from left to right, while the arrows are omitted for the sake of clarity.

results from previous calculations. This avoids many of the unnecessary computations of the PLVA.

Here, we present a recursive implementation of SLVA (Algorithm 4). The algorithm first allocates an array globally to store sequences of information (paths and costs) for all trellis vertexes. Then the algorithm visits the final state vertex and queries the ℓ th best path on it. When visiting a vertex $\sigma \in \Sigma_i(\mathcal{C})$, the algorithm is going to visit the two incoming vertexes ($\sigma \in \Sigma_{i-1}(\mathcal{C})$ and $\sigma - \mathbf{P}_i \in \Sigma_{i-1}(\mathcal{C})$) for the information and pick the better one, recursively. If an unallowable state vertex is reached, it returns a path with $+\infty$ cost for simplification. Finally, the ℓ th best path on the final state vertex is presented after a recursive search.

Algorithm 4 Serial list Viterbi Algorithm (SLVA)

Input: The parity-check matrix $\mathbf{P} \in \mathbb{F}_2^{(N-K) \times N}$, the reliability vector $|\mathbf{r}| \in \mathbb{R}^N$, the final state

$\mathbf{s}_{\text{final}} \in \mathbb{F}_2^{N-K}$, the rank of TEP to search ℓ .

Output: The ℓ th lightest TEP $\mathbf{e}^{(\ell)}$.

```

1:  $nodes \leftarrow$  an array of  $N \times |\mathbb{F}_2^{N-K}|$  sequences of  $(path, cost)$  pairs.  $\triangleright$  Allocate globally.
2: function SLVA( $\mathbf{P}, |\mathbf{r}|, \mathbf{s}_{\text{final}}, \ell$ )
3:    $\mathbf{s}_{\text{init}} \leftarrow \mathbf{0}$ .  $\triangleright$  Initial state.
4:   function SLVA-IMPL( $i, \mathbf{s}, \ell$ )  $\triangleright$  A recursive implementation.
5:     if  $i = 0$  then  $\triangleright$  Boundary of recursion.
6:       if  $\mathbf{s} = \mathbf{s}_{\text{init}}$  and  $\ell = 1$  then  $\triangleright$  Initial state  $\mathbf{s}_{\text{init}}$  has only 1 legal path.
7:         return  $(\varepsilon, 0)$ .  $\triangleright$  Empty string and initial cost.
8:       return  $(\varepsilon, +\infty)$ .  $\triangleright$  Infinity cost for unreachable states.
9:     while  $\text{length}(nodes[i][\mathbf{s}]) < \ell$  do  $\triangleright$  Check whether it is calculated or not.
10:      for  $b = 0, 1$  do  $\triangleright$  enumerate the incoming edges.
11:         $\mathbf{s}_b \leftarrow \mathbf{s} - b \cdot \mathbf{P}_i$ .  $\triangleright$  Minus operator is defined in vector space  $\mathbb{F}_2^{N-K}$ .
12:         $t_b \leftarrow$  the number of paths end with  $b$  in  $nodes[i][\mathbf{s}]$ .
13:         $(path_b, cost_b) \leftarrow$  SLVA-IMPL( $i - 1, \mathbf{s}_b, t_b + 1$ ).  $\triangleright$  Recursive search.
14:         $cost_b \leftarrow cost_b + b \cdot |r_i|$ .  $\triangleright$  Cost accumulation.
15:         $path_b \leftarrow path_b \parallel b$ .  $\triangleright$  Append a single bit.
16:         $b^* \leftarrow \arg \min_{b \in \{0,1\}} cost_b$ .  $\triangleright$  The optimal transition.
17:         $nodes[i][\mathbf{s}] \leftarrow nodes[i][\mathbf{s}] \parallel (path_{b^*}, cost_{b^*})$ .  $\triangleright$  Append to information sequence.
18:      return  $nodes[i][\mathbf{s}][\ell]$ .  $\triangleright$  The  $\ell$ th information of  $nodes[i][\mathbf{s}]$ .
19:    $(\mathbf{e}^{(\ell)}, cost) \leftarrow$  SLVA-IMPL( $N, \mathbf{s}_{\text{final}}, \ell$ ).
20: return  $\mathbf{e}^{(\ell)}$ .

```

REFERENCES

- [1] E. Berlekamp, R. McEliece, and H. Van Tilborg, "On the inherent intractability of certain coding problems (corresp.)," *IEEE Transactions on Information Theory*, vol. 24, no. 3, pp. 384–386, 1978.
- [2] M. 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.
- [3] C. Yue, M. Shirvanimoghaddam, Y. Li, and B. Vucetic, "Segmentation-discarding ordered-statistic decoding for linear block codes," in *2019 IEEE Global Communications Conference (GLOBECOM)*. IEEE, 2019, pp. 1–6.
- [4] C. Yue, M. Shirvanimoghaddam, G. Park, O.-S. Park, B. Vucetic, and Y. Li, "Linear-equation ordered-statistics decoding," *IEEE Transactions on Communications*, vol. 70, no. 11, pp. 7105–7123, 2022.
- [5] —, "Probability-based ordered-statistics decoding for short block codes," *IEEE Communications Letters*, vol. 25, no. 6, pp. 1791–1795, 2021.
- [6] Y. Wang, J. Liang, and X. Ma, "Local constraint-based ordered statistics decoding for short block codes," in *2022 IEEE Information Theory Workshop (ITW)*, 2022, pp. 107–112.
- [7] J. Liang, Y. Wang, S. Cai, and X. Ma, "A low-complexity ordered statistic decoding of short block codes," *IEEE Communications Letters*, vol. 27, no. 2, pp. 400–403, 2023.
- [8] H. Greenberger, "An efficient soft decision decoding algorithm for block codes," Jet Propulsion Laboratory, Pasadena, Calif., Tech. Rep., January and February 1979.
- [9] A. May, A. Meurer, and E. Thomae, "Decoding random linear codes in $O(2^{0.054n})$," in *Advances in Cryptology—ASIACRYPT 2011: 17th International Conference on the Theory and Application of Cryptology and Information Security, Seoul, South Korea, December 4-8, 2011. Proceedings 17*. Springer, 2011, pp. 107–124.
- [10] A. Becker, A. Joux, A. May, and A. Meurer, "Decoding random binary linear codes in $2^{n/20}$: How $1 + 1 = 0$ improves information set decoding," in *Advances in Cryptology—EUROCRYPT 2012: 31st Annual International Conference on the Theory and Applications of Cryptographic Techniques, Cambridge, UK, April 15-19, 2012. Proceedings 31*. Springer, 2012, pp. 520–536.
- [11] A. May and I. Ozerov, "On computing nearest neighbors with applications to decoding of binary linear codes," in *Advances in Cryptology—EUROCRYPT 2015: 34th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Sofia, Bulgaria, April 26-30, 2015, Proceedings, Part I*. Springer, 2015, pp. 203–228.
- [12] C. Yue, M. Shirvanimoghaddam, B. Vucetic, and Y. Li, "A revisit to ordered statistics decoding: Distance distribution and decoding rules," *IEEE Transactions on Information Theory*, vol. 67, no. 7, pp. 4288–4337, 2021.
- [13] A. Valembois and M. Fossorier, "A comparison between "most-reliable-basis reprocessing" strategies," *IEICE transactions on fundamentals of electronics, communications and computer sciences*, vol. 85, no. 7, pp. 1727–1741, 2002.
- [14] N. Seshadri and C.-E. Sundberg, "List Viterbi decoding algorithms with applications," *IEEE Transactions on Communications*, vol. 42, no. 234, pp. 313–323, 1994.
- [15] W. Lin, "Research on block Markov superposition transmission for ultra-reliable and low-latency communication," Master's thesis, Sun Yat-sen University, 2020.
- [16] Q. Wang, S. Cai, L. Chen, and X. Ma, "Semi-LDPC convolutional codes: Construction and low-latency windowed list decoding," *Journal of Communications and Information Networks*, vol. 6, no. 4, pp. 411–419, 2021.
- [17] M. Baldi, N. Maturo, E. Paolini, and F. Chiaraluce, "On the use of ordered statistics decoders for low-density parity-check codes in space telecommand links," *EURASIP Journal on Wireless Communications and Networking*, vol. 2016, no. 1, pp. 1–15, 2016.
- [18] S. Tang, S. Cai, and X. Ma, "A new chase-type soft-decision decoding algorithm for Reed-Solomon codes," *Alexandria Engineering Journal*, vol. 61, no. 12, pp. 13 067–13 077, 2022.
- [19] I. Tal and A. Vardy, "List decoding of polar codes," *IEEE Transactions on Information Theory*, vol. 61, no. 5, pp. 2213–2226, 2015.
- [20] A. Solomon, K. R. Duffy, and M. Médard, "Soft maximum likelihood decoding using grand," in *ICC 2020 - 2020 IEEE International Conference on Communications (ICC)*, 2020, pp. 1–6.
- [21] K. R. Duffy, J. Li, and M. Médard, "Capacity-achieving guessing random additive noise decoding," *IEEE Transactions on Information Theory*, vol. 65, no. 7, pp. 4023–4040, 2019.
- [22] X. Ma and A. Kavcic, "Path partitions and forward-only trellis algorithms," *IEEE Transactions on Information Theory*, vol. 49, no. 1, pp. 38–52, 2003.
- [23] G. Forney, "The viterbi algorithm," *Proceedings of the IEEE*, vol. 61, no. 3, pp. 268–278, 1973.

- [24] L. Bahl, J. Cocke, F. Jelinek, and J. Raviv, “Optimal decoding of linear codes for minimizing symbol error rate (corresp.),” *IEEE Transactions on Information Theory*, vol. 20, no. 2, pp. 284–287, 1974.
- [25] S. Lin, T. Kasami, T. Fujiwara, and M. Fossorier, *Trellises and trellis-based decoding algorithms for linear block codes*. Springer Science & Business Media, 1998.