Limited-Magnitude Error Correction for Probability Vectors in DNA Storage

Wenkai Zhang, Zhen Chen, Zhiying Wang
Center for Pervasive Communications and Computing (CPCC)
University of California, Irvine, USA
{wenkaiz1, zhenc4, zhiying}@uci.edu

Abstract—**DNA, with remarkable properties of high density and stability, particularly for long-term data archiving, is one of the most appealing storage media. Emerging DNA storage technologies use composite DNA letters, where information is represented by a probability vector, leading to higher information density and lower synthesizing cost than single DNA letters. However, it faces the problem of inevitable noise and information corruption. This paper studies the channel of composite DNA letters in DNA storage and block codes for symmetric limited-magnitude errors on probability vectors. We provide outer and inner bounds for limited-magnitude probability error correction codes. Moreover, we propose code constructions where the number of errors is bounded by t, the error magnitudes are bounded by l, and the probability resolution is fixed as k. Our constructions exploit the properties of the limited-magnitude errors, and improve the performance in terms of complexity and redundancy.**

I. Introduction

In the recent decade, DNA-based data storage systems are at the forefront of cutting-edge science and innovations. They are particularly appealing due to the high information density, in terms of physical volume, of DNA as compared to current state of the art storage media. Storing digital information on DNA involves encoding the information into a sequence over the DNA alphabet (that is, A, C, G and T), producing synthetic DNA molecules with the desired sequence, and storing the synthetic biological material in storage vessels. Reading the stored information requires sequencing of the DNA and decoding to obtain the original digital information [1]–[7].

However, DNA storage suffers from high cost, especially during the synthesis process. In order to break through the theoretical limit of 2 bits per synthesis cycle for single-molecule DNA, the use of composite DNA letters was introduced by Anavy et al. [8]. A composite DNA letter is a representation of a position in a sequence that constitutes a mixture of all four standard DNA nucleotides in predetermined (normalized) probabilities $\mathbf{x} = (x_A, x_C, x_G, x_T)$, where $x_A, x_C, x_G, x_T \in \mathbb{N}$, and $k = x_A + x_C + x_G + x_T$ is fixed to be the *resolution* parameter of the composite letter. For example, $\mathbf{x} = (3, 3, 3, 3)$ represents a position in a composite DNA sequence of resolution $k = 12$, in which there are 25%, 25%, 25% and 25% chances of seeing A, C, G and T, respectively. Multiple copies of the composite sequence are stored in order to realize the desired probabilities.

Writing a composite DNA letter in a given position of a DNA sequence is equivalent to producing (synthesizing) multiple copies (oligonucleotides) of the sequence. Thus, in this given position the different DNA nucleotides are distributed across the synthesized copies according to the specification of $\mathbf{x}$. Reading a composite letter requires the sequencing of multiple independent molecules representing the same composite sequence and inferring the original probabilities or composition from the observed base frequencies [8].

The inference at any fixed position is affected by the sequencing depth (number of times the position is read), as well as sequencing and synthesis errors, resulting in the probability change $\mathbf{x} = (x_A, x_C, x_G, x_T) \to \mathbf{y} = (y_A, y_C, y_G, y_T)$. The observed probabilities $\mathbf{y}$ and the the original probabilities $\mathbf{x}$ are usually close under correct operations and methods [8]. They also share the common resolution k. For example, the original probabilities are $\mathbf{x} = (3, 3, 3, 3)$, then after synthesising and sequencing, the observed probabilities are $\mathbf{x}' = (2, 4, 3, 3)$. The probability x_A is decreased by 1 and the probability x_C is increased by 1, keeping the sum $k = 12$. In general, errors change the composite probabilities of DNA letters in two directions (up or down) with limited magnitudes, called *limited-magnitude probability errors (LMPE)*. The illustration is shown in Figure 1.

We study block codes that correct limited-magnitude probability errors. A *symbol* includes the probabilities of the four standard DNA nucleotides. The errors are parameterized by two integer parameters: t is the number of erroneous symbols in a codeword, and l is the maximal magnitude of errors in one direction in a symbol, more specifically, the magnitudes of the upward changes and downward changes are both within l.

These errors are related to asymmetric errors over the set of integers modulo q. The case with binary alphabet is studied in [9]–[11]. Asymmetric limited-magnitude error-correcting codes are proposed in [12] for the case of correcting all symbol errors within a codeword. In [13], the authors study q-ary asymmetric limited-magnitude errors for multi-level flash. Systematic coding schemes with minimum redundant symbols for correcting q-ary asymmetric limited-magnitude errors are shown in [14] [15]. The work of [16] designed asymmetric Lee distance codes using binary encoding of the DNA alphabet and correct errors arising in solid state nanopore sequencing systems. To the best of our knowledge, the present work is

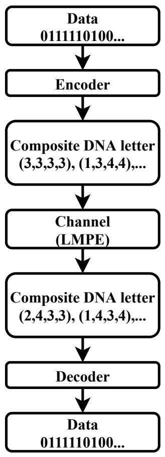

Fig. 1: The illustration of composite DNA storage

the first to consider limited-magnitude probability errors.

The main contributions of the paper are as follows. We model the errors in the composite DNA storage system as limited-magnitude probability errors. Moreover, we present the sphere packing bound and Gilbert-Varshamov bound for the family of limited-magnitude probability error correction codes. Code constructions are provided which first partition the probability tuples into classes by taking advantage of the error characteristics, and then protect the classes with non-binary error correction codes. The resulting codes are over a smaller finite field size compared to naive error correction on the entire symbols, leading to lower computation complexity (see, e.g., [17]).

Notation. Vectors are denoted by bold letters. For a positive integer i, denote by $[i]=\{1,2,\dots,i\}$. For two integers $i\le j$, denote by $[i,j]=\{i,i+1,\dots,j\}$. C_n^k denotes the binomial coefficient n choose k. For positive integers i,j, write $b=i \mod j$ as the remainder after i is divided by j, where $0\le b\le j-1$. The notation $b\equiv i \mod j$ means that integers b and i are congruent modulo j.

II. Problem statement

We consider a problem of protecting information when it faces the limited-magnitude probability errors. The word $\mathbf{x}=(\mathbf{x}_1,\mathbf{x}_2,\dots,\mathbf{x}_N)$ has N symbols, and each symbol or probability tuple $\mathbf{x}_i=(x_{i1},x_{i2},\dots,x_{im}), i\in[N]$, has m (normalized) probability values satisfying

$$0\le x_{ij}\le k, j\in[m], \tag{1}$$

$$\sum_{j=1}^{m} x_{ij}=k, \tag{2}$$

where k is the fixed resolution parameter to keep the summation of probabilities to 1, namely, $\sum_{j=1}^{j=m}\frac{\mathbf{x}_{ij}}{k}=1$. The set of all words of length N satisfying (1) and (2) are denoted by $\mathcal{X}$. One can easily check that its size is $|\mathcal{X}|=\left(C_{k+m-1}^{m-1}\right)^N$.

Assume $\mathbf{x}\in\mathcal{X}$ is the transmitted codeword. Denote by $\mathbf{y}=(\mathbf{y}_1,\mathbf{y}_2,\dots,\mathbf{y}_N)\in\mathcal{X}$ the received word. Denote by $\mathbf{e}=\mathbf{y}-\mathbf{x}=(\mathbf{e}_1,\mathbf{e}_2,\dots,\mathbf{e}_N)$ the error vector, and $e_{i,j}$ its probability difference values, $i\in[N], j\in[m]$. It is apparent that for all $i\in[N]$,

$$\sum_{j=1}^{m} e_{i,j}=0. \tag{3}$$

The errors of interest are introduced in Definition 1.

Definition 1 (Limited-magnitude probability error (LMPE)): A limited-magnitude probability error $\mathbf{e}$ is called an (l,t) LMPE if the number of symbol errors is at most t, and the error magnitude of each symbol is at most l. Formally,

$$|\{i\in[N]:\mathbf{e}_i\ne\mathbf{0}\}|\le t, \tag{4}$$

$$\sum_{j\in[m]:e_{ij}>0} e_{ij}\le l, \forall i\in[N]. \tag{5}$$

Equation (5) indicates that the upward error terms sum to at most l. Due to (3), the downward error terms also sum to at most l.

For this paper, we consider the case of $m=4$, which fits the composite DNA storage application.

For example, when $k=12$, the transmitted word is $\mathbf{x}$=$((3,3,3,3),(2,4,3,3),(1,7,2,2))$, the received word is $\mathbf{x}'$=$((1,3,4,4),(1,5,3,3),(1,7,2,2))$. Here, the number of errors is $t=2$ and the error magnitude is $l=2$.

For a particular symbol, an *l-limited-magnitude error* $\mathbf{e}=(e_1,e_2,e_3,e_4)$ is defined similar to (5) as

$$\sum_{j\in[4]:e_j>0} e_j\le l, \qquad \sum_{j\in[4]:e_j<0}|e_j|\le l. \tag{6}$$

Lemma 1: $\mathbf{e}=(e_1,e_2,e_3,e_4)$ is an l-limited-magnitude error for one symbol, if and only if

$$\sum_{j=1}^{4}|e_j|\le 2l. \tag{7}$$

Proof: First, note that $\sum_{j=1}^4 e_j=0$. Equivalently,

$$\sum_{j\in[4]:e_j>0} e_j=\sum_{j\in[4]:e_j<0}|e_j|. \tag{8}$$

If $\mathbf{e}$ is an l-limited-magnitude error, then summing up the two inequalities in (6) gives (7).

If $\mathbf{e}$ is an error with magnitude more than l, then at least one inequality in (6) is violated. Without loss of generality, assume the first one is violated. Then,

$$l<\sum_{j\in[4]:e_j>0} e_j=\sum_{j\in[4]:e_j<0}|e_j|. \tag{9}$$

Hence, $\sum_{j=1}^4|e_j|>2l$ as desired. ■

An (l,t) *LMPE correction code* is defined as an encoding function $Enc:\{0,1\}^K\to\mathcal{X}$ and a decoding function $Dec:\mathcal{X}\to\{0,1\}^K$. Here, K is the information length in bits, and N is the codeword length in symbols. For any binary information vector $\mathbf{u}$ of length K, and any (l,t) LMPE vector $\mathbf{e}$ of length N, the functions should satisfy

$$Dec(Enc(\mathbf{u})+\mathbf{e})=\mathbf{u}. \tag{10}$$

III. Bounds on LMPE correction codes

In this section, we derive sphere packing bound and Gilbert-Varshamov bound on the code size for correcting (l,t) LMPE. To that end, we define a distance that captures the capability of an LMPE correction code.

Definition 2 (Distance): Let l be a fixed integer. Consider a graph whose vertices are the set of words in $\mathcal{X}$. Two vertices $\mathbf{x},\mathbf{z}$ are connected by an edge if their difference $\mathbf{e}=\mathbf{z}-\mathbf{x}$ is an $(l,t=1)$ LMPE. The distance $d_l(\mathbf{x},\mathbf{z})$ of any two words $\mathbf{x},\mathbf{z}$ is defined as the length of the shortest path between them.

The definition above is indeed the geodesic distance, which is a distance metric.

Remark 1: Note that if a codeword $\mathbf{x}$ is transmitted and an (l,t) LMPE occurs, then the received word $\mathbf{y}$ must satisfy $d_l(\mathbf{x},\mathbf{y})\leq t$. On the other hand, if $d_l(\mathbf{x},\mathbf{y})=t$, the error may not be an (l,t) LMPE, because there can be less than t symbol errors but some symbol error can have a magnitude more than l.

The distance gives a sufficient condition for the number of limited-magnitude symbol errors correctable by a code $\mathcal{C}$, for a fixed error magnitude l.

Proposition 1: A code $\mathcal{C}$ can correct an (l,t) LMPE if $d_l(\mathbf{x},\mathbf{z})\geq 2t+1$ for all distinct $\mathbf{x},\mathbf{z}\in\mathcal{C}$.

Proof: Assume the code has minimum distance $2t+1$, but an (l,t) LMPE is not correctable. Then there exist two codewords $\mathbf{x},\mathbf{z}$ and a received word $\mathbf{y}$ such that $d_l(\mathbf{x},\mathbf{y})\leq t, d_l(\mathbf{z},\mathbf{y})\leq t$. Therefore, $d_l(\mathbf{x},\mathbf{z})\leq d_l(\mathbf{x},\mathbf{y})+d_l(\mathbf{z},\mathbf{y})\leq 2t$, which is a contradiction to the minimum distance $2t+1$. ■

Let $A(N,l,t)$, denote the maximum possible size of an (l,t) LMPE correction code, whose codeword length is N. The outer and inner bounds on the code size are given next. Detailed derivations can be found in the full version [18].

Sphere packing bound. The sphere packing bound states that the code size is upper bounded by the size of all words $\mathcal{X}$ divided by the size of the error ball, which is the number of possible received words for a codeword. If a probability value is smaller than l (or larger than $k-l$), the downward (or upward) error magnitude becomes smaller than l as well, resulting in different error ball sizes. To accommodate the issue, we relax the set of words to be

$$\overline{\mathcal{X}}=\{\mathbf{x}+\mathbf{e}: \text{for all } \mathbf{x}\in\mathcal{X} \text{ and } (l,t) \text{ LMPE } \mathbf{e}\}. \tag{11}$$

As a result, a "probability" value in $\overline{\mathcal{X}}$ can be negative or more than k. The sphere packing bound, for $k\gg l$, can be approximately represented as:

$$A(N,l,t)\leq\frac{\sum_{t'=0}^{t}C_N^{t'}(C_{k+3}^3)^{N-t'}(2k^2l)^{t'}}{\sum_{t'=0}^{t}C_N^{t'}(\frac{10}{3}l^3+5l^2+\frac{11}{3}l)^{t'}+1}. \tag{12}$$

Gilbert-Varshamov bound. By Proposition 1, the distance being at least $2t+1$ is a sufficient condition to correct (l,t) LMPE. Therefore, Gilbert-Varshamov bound implies that the code size is lower bounded by the size of $\mathcal{X}$ divided by the size of a ball of radius $2t+1$, where the ball is defined as the set of words whose distance is at most $2t+1$ from the center. Note that this ball of radius d is larger than the error ball of d LMPE used in the sphere packing bound. For sufficiently large N and $l\gg\frac{3}{2}$, we can approximately obtain

$$A(N,l,t)>\frac{(C_{k+3}^3)^N}{(2t+1)C_N^{2t+1}\left(\frac{10}{3}\right)^{2t+1}l^{3(2t+1)}}. \tag{13}$$

IV. Constructions

In this section, we first use an example to illustrate the coding problem and introduce the main idea of the code constructions. Then we give the framework of the proposed code for the limited-magnitude probability errors. Based on the framework, we provide three code constructions.

Our first example makes use of Hamming codes, which are perfect 1-error correcting codes. For the sake of completeness, we briefly review q-ary Hamming code [19], [20] below.

Definition 3 (Hamming code): $(\frac{q^r-1}{q-1},\frac{q^r-1}{q-1}-r)$ Hamming code over $GF(q)$ has codeword length $\frac{q^r-1}{q-1}$, information length $\frac{q^r-1}{q-1}-r$ and r redundant symbols. The Hamming code is defined by its parity check matrix of size $r\times\frac{q^r-1}{q-1}$, whose columns are all non-zero columns of length r and pair-wise independent.

Example 1: The design goal is to protect the probabilities against $(l=1,t=1)$ LMPE. The resolution parameter for the DNA letters is set as $k=12$. The symbol alphabet size is $C_{k+3}^3=455$. We have N symbols and each symbol has four probability values in the range from 0 to k. For every symbol, each probability value x_j is divided by $2l+1=3$, and can be represented by the quotient (a_j) and remainder (b_j), for $j\in[4]$.

$$x_j=3a_j+b_j,\quad j\in[4]. \tag{14}$$

When $k=12$, the quotient tuple for a symbol is in the set $\{(a_1,a_2,a_3,a_4):\sum_{j=1}^4 a_j\leq 4, a_j\in[0,4], j\in[4]\}$, and the remainder tuple belongs to the set $\{(b_1,b_2,b_3,b_4):\sum_{j=1}^4 b_j\in\{0,3,6\}, b_j\in[0,2], j\in[4]\}$.

When the errors occur, the remainders and the quotients will be changed, but we can recover the transmitted symbol based on the correct remainder tuple and the received symbol. Consider the example in Figure 2, the second transmitted symbol is $(3,3,3,3)$. Then $(l=1,t=1)$ LMPE occurs, and the received symbol is $(2,4,3,3)$, whose remainder tuple is $(2,1,0,0)$. Based on the remainder correction codes which will be introduced later, we can get the correct remainder tuple $(0,0,0,0)$. Then we know that the first and second probabilities $(2,4)$ have errors and should be corrected as $(3,3)$ since the error magnitude is $l=1$.

For the remainder tuples, we have 27 different possibilities, shown as Table I. The remainder tuples are mapped arbitrarily to elements in $GF(27)$. So the remainder tuples can be protected by a 27-ary Hamming code. Figure 2 uses the 27-ary $(28,26)$ Hamming code with $r=2$ parities. In the last two symbols, the remainders are parities, and their quotients can still be used as information.

We note that since $k=12$, the quotients and the remainders are not independent. When $\sum_{j=1}^4 b_j=6$, we must

TABLE I: The mapping between remainders and $GF(27)$. Each element of $GF(27)$ is denoted by an integer between 0 and 26.

remainder	Element	remainder	Element	remainder	Element
0000	0	1110	1	2220	2
0111	3	1221	4	2001	5
0222	6	1002	7	2112	8
0012	9	1122	10	2202	11
0021	12	1101	13	2211	14
0210	15	1020	16	2100	17
0102	18	1212	19	2022	20
0201	21	1011	22	2121	23
0120	24	1200	25	2010	26

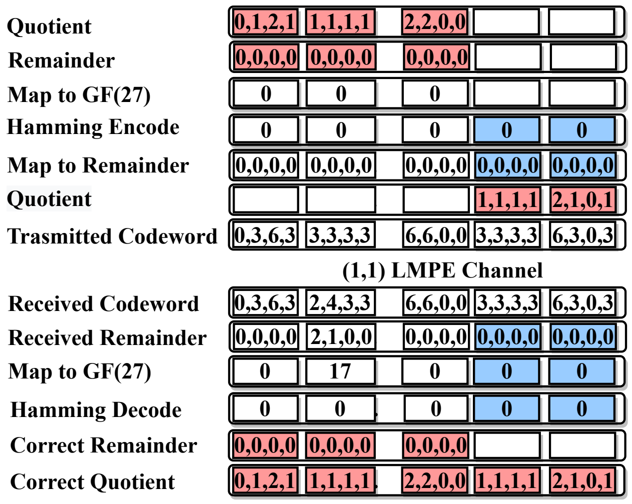

Fig. 2: An $(l = 1, t = 1)$ LMPE correction code using the 27-ary $(28, 26)$ Hamming code. Assume $(0, 0, 0, 0)$ is mapped to 0 in $GF(27)$. The red tuples are the information part, and the blue ones are the parities.

have $\sum_{j=1}^{4} a_j = 2$. And in this case we have the least number of possible quotient tuples, which is 10. Given the first 26 information symbols, the last 2 parity remainder tuples will be fixed. In order to accommodate the worst-case scenario, we only allow 10^2 possible information messages represented by the last two quotient tuples. For example, in Figure 2, the last two quotient tuples $(1, 1, 1, 1), (2, 1, 0, 1)$ correspond to one of these 10×10 messages. The rate is $(26 \log_2 455 + \log_2 10^2)/(28 \log_2 455) = 0.955$.

Our method of correcting errors only on the remainders is inspired by [13], where the modulo operation was introduced for limited magnitude errors on integers modulo q.

In this example, we view symbols with the same remainder tuple as the same "class". Hamming code is employed to protect the classes. Motivated by the example, we design two-layer codes to correct limited-magnitude probability errors, shown in the following framework.

Construction 1 (Framework):

Symbol classification. Symbols are mapped to classes such that the l-limited-magnitude symbol error always changes the class of the symbol.

First layer. Construct a t-error correction code whose codeword symbols are the class indices. Now the locations of erroneous symbols can be identified.

Second layer. Construct a code such that given the correct classes and the received symbols, the original symbols can be recovered.

Based on this framework, we can provide different coding schemes to protect the information in the limited-magnitude probability error channel.

While any t-error correction code over the class alphabet suffices for the first layer, we illustrate our ideas using the well-known BCH codes. BCH codes include Hamming codes as a special case. For a given field $GF(q)$ and an integer m, q-ary BCH code has codeword length $q^m - 1$, minimum distance at least $2t + 1$, and at most $2mt$ parity check symbols.

The next construction is a generalization of Example 1.

Construction 2 (Remainder classes):

Reduce each probability value modulo $(2l + 1)$, and then classify a symbol by the remainder tuple. The number of classes is $(2l + 1)^3$, since the sum of the remainders must be congruent to k modulo $2l + 1$. Then we use BCH code over a field of size at least $(2l + 1)^3$ with distance $2t + 1$ in the first layer. The second layer is the identity code.

Noting that it is possible to include multiple remainder tuples in the same class if the symbols cannot be converted to each other by l-limited-magnitude errors. We propose a classification with fewer classes and less complexity, for $l \leq 4$.

Similar to Construction 2, we first reduce the probabilities modulo $(2l+1)$ and obtain $(2l+1)^3$ possible remainder tuples. Next, we will further partition them into $(2l + 1)^2$ classes, each with $(2l + 1)$ remainder tuples. The valid classification is required to satisfy two conditions:

C1. The difference of any two symbols in the same class is not a l-limited magnitude error.

C2. Each of the $(2l + 1)^3$ remainder tuples is included in exactly one class.

Notations. For a remainder tuple $\mathbf{b} = (b_1, b_2, b_3, b_4)$ and an integer i, we denote

$$i\mathbf{b} = (ib_1, ib_2, ib_3, ib_4) \mod (2l + 1) \qquad (15)$$

as the scaled remainder tuple. For two remainder tuples $\mathbf{b}, \mathbf{c}$, we write

$$\mathbf{b} + \mathbf{c} = \mathbf{b} + \mathbf{c} \mod (2l + 1) \qquad (16)$$

as the sum remainder tuple. For limited magnitude l, a remainder tuple $\mathbf{d}$, whose sum is 0 modulo $2l + 1$, is said to be a *remainder error pattern* if

$$\sum_{j \in [4]: d_j \in [0,l]} d_j + \sum_{j \in [4]: d_j \in [l+1, 2l]} (2l + 1 - d_j) \leq 2l. \qquad (17)$$

Let $\mathbf{b}$ be a remainder tuple where $b_1 = 1$, $\sum_{j=1}^{4} b_j \mod (2l+1) = 0$, and $i\mathbf{b}$ is not a remainder error pattern for any $i \in [2l]$. Then $\mathbf{b}$ is called a *critical tuple.*

The following proposition provides a valid classification.

Proposition 2: If $\mathbf{b}$ is a critical pattern, then the classification in Table II is valid.

Proof: We need to prove that Conditions C1 and C2 are satisfied.

TABLE II: Classification for limited magnitude l. Column 0 lists all remainder tuples where the first entry is 0 and the sum is 0 modulo $2l+1$. In this table, $k \mod (2l+1) = 0$. For other k, the last entry in each remainder tuple is added by k modulo $2l+1$. For instance, if $k = 14, l = 1$, $(0,0,1,2l)$ in Class 1 and Column 0 becomes $(0,0,1,2l+k) \mod 3 = (0,0,1,1)$.

Class	Column 0	Column 1	...	Column $2l$
0	$(0,0,0,0)$	$\mathbf{b}$	...	$2l\mathbf{b}$
1	$(0,0,1,2l)$	$(0,0,1,2l)+\mathbf{b}$	...	$(0,0,1,2l)+2l\mathbf{b}$
...	...	...	...	...
$(2l+1)^2$	$(0,2l,2l,2)$	$(0,2l,2l,2)+\mathbf{b}$	...	$(0,2l,2l,2)+2l\mathbf{b}$

TABLE III: Critical tuples for $l \leq 4$.

l	Critical tuple
1	$(1,1,1,0)$
2	$(1,1,2,1)$
3	$(1,2,3,1)$
4	$(1,4,6,7)$

C1. The remainder tuples in each class are in the form of $\mathbf{c} + i\mathbf{b}$, $0 \leq i \leq 2l$, where $\mathbf{c}$ is the tuple in Column 0 of the table. Therefore, for two symbols in a class, the difference of their remainder tuples must be $i\mathbf{b}$, for some $0 \leq i \leq 2l$. Assume C1 does not hold. Then there exist two symbols in the same class, such that their difference $\mathbf{e} = (e_1, e_2, e_3, e_4)$ is an l-limited-magnitude error. Let $i\mathbf{b} = (d_1, d_2, d_3, d_4)$ be the difference of their remainder tuples. It can be checked that since $|e_j| \leq l$ for $j \in [4]$,

$$|e_j| = \begin{cases} d_j, & \text{if } d_j \in [0,l], \\ 2l+1-d_j, & \text{if } d_j \in [l+1,2l]. \end{cases} \tag{18}$$

Therefore,

$$\sum_{j\in[4]:d_j\in[0,l]} d_j + \sum_{j\in[4]:d_j\in[l+1,2l]} (2l+1-d_j) \tag{19}$$

$$= \sum_{j=1}^{4} |e_j| \leq 2l, \tag{20}$$

where the last inequality follows by Lemma 1. Now there is a contradiction to the definition of the critical pattern, where $i\mathbf{b} = (d_1, d_2, d_3, d_4)$ must not satisfy (17).

C2. Any tuple in the 0-th column sums to k modulo $(2l+1)$. Adding $i\mathbf{b}$ does not change the sum of the remainder tuple modulo $(2l+1)$. Therefore, all tuples listed in the table sums to k modulo $(2l+1)$, as desired. Since the 0-th column lists all remainder tuples where the first entry is 0, and the first entry of $\mathbf{b}$ is 1, adding $i\mathbf{b}$ gives all remainder tuples where the first entry is i in the i-th column, for $1 \leq i \leq 2l$. Hence, each remainder tuple is listed in exactly one class. ■

A critical tuple $\mathbf{b}$ is found by Algorithm 1 for $l \leq 4$. We list one critical tuple for each l, $l \leq 4$, in Table III. As the error magnitude l increases, the number of remainder error patterns becomes larger. When $l > 4$, the proportion of remainder error patterns in all the remainder tuples is higher than 50%. Hence, the classification method is suitable when $l \leq 4$.

Now we are ready to describe the code construction using the above classification.

Construction 3 (Reduced classes, $l \leq 4$):

Algorithm 1: Find a critical tuple.

```
Input: Error magnitude limit $l$
Output: A critical tuple $\mathbf{b}$
1 for all $\mathbf{b}$ whose sum is 0 modulo $2l+1$ and $b_1 = 1$ do
2     for $i = 1$ to $2l$ do
3         if $i\mathbf{b}$ is a remainder error pattern then
4             break;
5     if $\mathbf{b}$ is found then
6         return;
```

For the first layer, calculate the remainder tuples of the symbols and classify them as in Table II. Use BCH code with distance $2t+1$ over a field of size at least $(2l+1)^2$ to protect the class indices. Note that the erroneous symbols are now identified and they can be treated as erasures. The first entry in the remainder tuple determines the remainder tuple for a given class. In the second layer, for the first entry of every remainder tuple in the codeword, apply BCH code with distance $t+1$ over a field of size at least $2l+1$ to correct t erasures.

The following construction is an improvement over Construction 2 for $(l = 1, t = 1)$ LMPE, inspired by the work on efficient non-binary Hamming codes for limited-magnitude errors on integers [21]. The main idea is that for different transmitted remainder class indices in $GF(27)$, their erroneous class indices do not contain all possible elements in $GF(27)$, and their erroneous class indices are the same. Thus, an improved Hamming code over $GF(27)$ can be applied to the class indices. The improved code benefits from a larger codeword length while sharing the same redundancy as Hamming code. Details of the parity check matrix construction of the improved code is in the full version [18].

Construction 4 (Improved Hamming code, $l = 1, t = 1$): For the first layer, map symbols according to the remainder tuples to $GF(27)$ in the same way as Construction 2. Given the remainder error patterns for $l = 1$, the parity check matrix of the improved Hamming code can be formed. One example of systematic parity check matrix with $r = 2$ redundancies is as below:

$$\begin{bmatrix} 0 & 1 & \dots & 1 & 7 & \dots & 7 & 1 & 0 \\ 7 & 1 & \dots & 26 & 0 & \dots & 26 & 0 & 1 \end{bmatrix}, \tag{21}$$

where the integers denote the exponent in the power representation of $GF(27)$, and the primitive polynomial x^3+2x+1 is used to represent elements in $GF(27)$. Thus we get a $(56, 54)$ Hamming code, which has a better rate than the original $(28, 26)$ Hamming code. In general, the codeword length is $N = \frac{1}{13}(27^r - 1)$, for redundancy r. The second layer is the identity code.

V. Comparisons

In this section, different code constructions are compared. As illustrative examples, we first consider the codes for $(l = 1, t = 1)$ LMPE. Then we compare the performance

TABLE IV: Comparison for the proposed constructions.

Method	Error	Field size	Redundancy in bits
Naive Hamming code	$l=1,t=1$	C_{k+1}^3	$\log_2\left(\left(C_{k+3}^3-1\right)N+1\right)$
Hamming code with remainder classes	$l=1,t=1$	27	$\log_2(26N+1)$
Improved Hamming with remainder classes	$l=1,t=1$	27	$\log_2(13N+1)$
Hamming code with reduced classes	$l=1,t=1$	9	$\log_2(8N+1)+\log_2(3)$
BCH code with remainder classes	l,t	$(2l+1)^3$	$2t\log_2(N+1)$
BCH code with reduced classes	$l\le 4,t$	$(2l+1)^2$	$3t\log_2(N+1)$

when there are t errors. For the criteria, we compare the field size and hence the computational complexity, as well as the redundancy in bits with fixed codeword length N.

The results are in Table IV whose derivation is in [18]. In the table, naive Hamming code refers to the single-error correction code applied to the entire probability symbol, requiring a field of size of at least C_{k+3}^3.

The improved Hamming code has the least redundant bits when there is a single error with limited magnitude $l=1$. When there are t errors with limited magnitude $l\le 4$, BCH code with remainder classes has less redundancy than BCH code with reduced classes. The reason is that the number of redundant bits in BCH code remains the same irrespective of the field size. But the reduced classes require an extra coding in the second layer, leading to a worse code rate. However, since the encoding/decoding operations are over smaller fields, the reduced classes benefit from lower complexity.

VI. Conclusion

This paper proposes a new channel model motivated by composite DNA storage. Different from traditional channels, the symbols are probability vectors and the channel noise is modeled as limited-magnitude errors. We develop a framework of two-layer error correction codes, where symbols are classified and then encoded. An attractive property of the classification herein, is that it takes advantage of the error characteristics, and hence reduces the computation complexity. Other classification methods and codes for specific probability error patterns are interesting future directions. Our model can also be applied to other problems where the information can be represented by probability distributions.

Appendix A
Sphere packing bound

Here, we provide the derivation of the sphere packing bound. Our discussion focuses on the case with large k, or a large resolution. We will use the following two formulas: the number of non-negative choices to have $x_1+x_2+\cdots+x_r=k$ is C_{k+r-1}^{r-1}, and the number of positive choices to have $x_1+x_2+\cdots+x_r=k$ is C_{k-1}^{r-1}.

The actual set of possible symbols or probability vectors is of size (C_{k+3}^3), where each probability is between 0 and k. To use the sphere packing bound, we adopt the extended alphabet $\overline{\mathcal{X}}$ as in (11), the virtual received probability is corrupted by the limited-magnitude error and is assumed to be in the arrange $-l$ to $k+l$. We consider the extended symbol alphabet $\{(x_1,x_2,x_3,x_4)+(e_1,e_2,e_3,e_4): x_j\in[0,k], j=1,2,3,4, \sum_{j=1}^4 x_j=1, \sum_{i=1}^4[e_j]^+\le l\}$. There are 3 cases where the probabilities in a symbol can be negative (and more than k).

(i) There is one negative probability $-i$, for $i\in[l]$, and the rest three non-negative probabilities sum up to $k+i$. The number of possibilities is

$$C_4^1\sum_{i=1}^{l}C_{k+i+2}^2. \tag{22}$$

(ii) There are two negative probabilities whose sum is $-i$, for $i\in[l]$, and the rest two non-negative probabilities sum up to $k+i$. The number of possibilities can be computed as:

$$C_4^2\sum_{i=1}^{l}C_{i-1}^1C_{k+i+1}^1 \tag{23}$$

(iii) There are three negative probabilities whose sum is $-i$, for $i\in[l]$, and the remaining probability is $k+i$. Then The number of possibilities can be expressed as

$$C_4^3\sum_{i=1}^{l}C_{i-1}^2. \tag{24}$$

After summing up the three cases, the number of possibilities for symbols with negative probabilities can be represented as

$$2k^2l+3kl^2+8kl+k+5l^3+13l^2-3l+13. \tag{25}$$

When $k\gg l$, it can be approximated by

$$2k^2l. \tag{26}$$

We consider $\overline{\mathcal{X}}$ such that there are at most t errors with maximum magnitude l in words of length N, where negative probabilities are allowed. There are at most t symbols with negative probabilities, hence

$$|\overline{\mathcal{X}}|=\sum_{t'=0}^{t}C_N^{t'}(C_{k+3}^3)^{N-t'}(2k^2l)^{t'} \tag{27}$$

On the other hand, let E be the number of possible errors of maximum magnitude l, then the error ball centered at a given word is of size

$$|\text{error ball}|=\sum_{t'=0}^{t}C_N^{t'}E^{t'}+1. \tag{28}$$

Similar to the calculation of the extended alphabet size, the expression of E includes three cases for 1, 2, or 3 positive upward errors (correspondingly 3, 2, or 1 non-negative downward errors, respectively), and

$$E = \sum_{i=1}^{l} \left(C_4^1 C_{i+2}^2 + C_4^2 C_{i-1}^1 C_{i+1}^1 + C_4^3 C_{i-1}^2 \right), \tag{29}$$

$$= \frac{10}{3} l^3 + 5l^2 + \frac{11}{3} l. \tag{30}$$

Here, i denotes the sum of the upward or downward error magnitude.

Now the sphere packing bound approximately can be calculated as follows.

$$A(N, l, t) \leq \frac{|\overline{\mathcal{X}}|}{|\text{error ball}|} \tag{31}$$

$$\approx \frac{\sum_{t'=0}^{t} C_N^{t'} (C_{k+3}^3)^{N-t'} (2k^2 l)^{t'}}{\sum_{t'=0}^{t} C_N^{t'} E^{t'} + 1}, \tag{32}$$

where the approximation holds when $k \gg l$. For general k, l, the bound can be calculated by using (25) for the extended alphabet size.

Ignoring lower order terms, when $N \gg k$, the bound is in the order of

$$\frac{1}{6^{N-t}} \left(\frac{3}{10} \right)^t \frac{k^{3N-t}}{l^{3t}}. \tag{33}$$

When $N \ll k$, meaning the resolution is higher than the codeword length, the bound is in the order of

$$\frac{1}{C_N^t} \frac{1}{6^N} \left(\frac{3}{10} \right)^t \frac{k^{3N}}{l^{3t}}. \tag{34}$$

In the latter case, the extended alphabet $\overline{\mathcal{X}}$ has almost the same size as the original alphabet $\mathcal{X}$.

Appendix B
Gilbert-Varshamov bound

In this section, we give the insight of Gilbert-Varshamov bound. We consider the ball of radius $d = 2t + 1$ whose center is a codeword. It is implicitly assumed that the distance is defined for error magnitude l. This ball corresponds to d different error cases. The i-th case is that i symbols have errors whose sum maximum magnitude is dl, and each symbol error maximum magnitude is a multiple of l.

When the magnitude of error $l = l'$ is fixed, the possible error patterns E is to replace l with l' in (30). When $l' \gg \frac{3}{2}$, we approximate the possible error patterns as $E \approx \frac{10}{3} l'^3$. So the volume of the i-th case (V_i) can be represented as:

$$V_i = C_N^i \sum_{(l_1, l_2, \ldots, l_i)} \left(\frac{10}{3} \right)^i l_1^3 \ldots l_i^3, \tag{35}$$

where the parameter l_j is the maximum magnitude of the j-th error, for $j \in [i]$. The summation is over all tuples $(l_1, l_2, \ldots, l_i)$ such that each l_j is a positive multiple of l, and $\sum_{j=1}^{i} l_j = ld$. The number of such tuples is C_{d-1}^{i-1}. Based on the inequality of arithmetic and geometric means, we have the following inequality:

$$(l_1 l_2 \ldots l_i) \leq \frac{l_1 + l_2 + \cdots + l_i}{i})^i, \tag{36}$$

with equality if and only if $l_r = l_w, r \neq w, r, w \in [i]$. Then the upper bound of V_i, denoted by $U(i)$ can be represented as:

$$U(i) = C_N^i C_{d-1}^{i-1} \left(\frac{10}{3} \right)^i \left(\frac{dl}{i} \right)^{3i} \tag{37}$$

Next, we have another the inequality of lower bound.

$$l_1 l_2 \ldots l_i \geq (d - i + 1) l^i, \tag{38}$$

with equality if and only if $l_j = l, j \in [i-1], l_i = (d-(i-1))l$. So the lower bound of V_i, denoted by $L(i)$, is:

$$L(i) = C_N^i C_{d-1}^{i-1} \left(\frac{10}{3} \right)^i (d - i + 1)^3 l^{3i}. \tag{39}$$

Then we consider the following ratio:

$$\frac{L(i+1)}{U(i)} = \frac{10}{3} \frac{N - i}{i(i+1)} (d - i)^4 l^3 \left(\frac{i}{d} \right)^{3i}, \tag{40}$$

where $i \in [d - 1]$. It can be seen that (40) is larger than 1, when N is large, more specifically, when

$$N > \frac{3}{10 l^3} \frac{i(i+1)}{(d-i)^4} \left(\frac{d}{i} \right)^{3i}. \tag{41}$$

Here, $\frac{i(i+1)}{(d-i)^4} \geq d^2 - d$ with equality when $i = d - 1$. And $\left(\frac{d}{i} \right)^{3i} \geq e^{\left(\frac{3d}{e} \right)} = 3.02^d$ with equality when $i = \frac{d}{e}$. Thus, we require that $N = O(\frac{1}{l^3} d^2 3.02^d)$. In this case, the volume of the ball with radius d is upper bounded as

$$\sum_{i=1}^{d} V_i < dV_d = dC_N^d \left(\frac{10}{3} \right)^d l^{3d}. \tag{42}$$

In fact, if $N \gg \frac{1}{l^3} d^2 3.02^d$, the volume of the ball is dominant by the term V_d, and all other terms can be neglected. This means that the error ball is almost the same as the ball with (l, t) LMPE, despite the difference between the LMPE and the distance as in Remark 1.

So for $N = O(\frac{1}{l^3} d^2 3.02^d), l \gg \frac{3}{2}$, the Gilbert-Varshamov bound can be roughly expressed to be:

$$A(N, l, t) \geq \frac{|\mathcal{X}|}{|\text{ball of radius } 2t + 1|} \tag{43}$$

$$> \frac{(C_{k+3}^3)^N}{(2t + 1) C_N^{2t+1} \left(\frac{10}{3} \right)^{2t+1} l^{3(2t+1)}}. \tag{44}$$

When l is small and N is sufficiently large, the above bound can be modified by applying the exact formula of E in (30). Ignoring the lower order terms, the bound is in the order of

$$\frac{(2t)!}{N^{2t+1}} \frac{1}{6^N} \left(\frac{3}{10} \right)^{2t+1} \frac{k^{3N}}{l^{3(2t+1)}} \tag{45}$$

Appendix C
Improved Hamming code

We provide the method of [21] to choose the parity check matrix H for Hamming code when not all error patterns are possible as in Construction 4.

Notations. The major element is defined as the leading non-zero element in a column of the parity check matrix, shown as red numbers on Figure 3. Major columns are defined as the columns whose major element is 1. A minor column is defined as a column whose major element is not 1. Note that parity check matrix of Hamming code can be constructed by including all major columns. Each element in $GF(q)$ is denoted by an integer between 0 and $q-1$. For $i, j \in GF(q)$, we say $i < j$ if the integer representation of i is smaller than that of j. Let $\mathcal{E}$ be the set of possible error patterns. For example, for $(l = 1, t = 1)$ LMPE the remainder error patterns are listed in Table V.

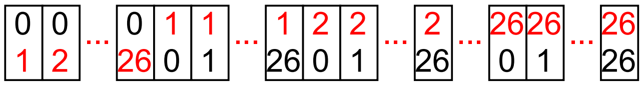

Fig. 3: Major elements in all possible columns for $GF(27)$ and 2 parity check symbols.

The following construction is the procedure to find the parity check matrix of the improved Hamming code based on [21]. The idea is to add columns to the parity check matrix of Hamming code, so that the redundancy r is kept but the codeword length N is enlarged. For the sake of completeness, we prove its correctness in Proposition 3.

Construction 5 (Parity check matrix of the improved Hamming code): Initialize the parity check matrix H by including all major columns over $GF(q)$ of length r. Initialize the set $\mathcal{I} = \{1\}$. For each $i \in GF(q)\backslash\{0, 1\}$, append to H minor columns with major element i if the following is satisfied: for all $e_1, e_2 \in \mathcal{E}$ and $j \in \mathcal{I}$,

$$ie_1 \neq je_2. \tag{46}$$

Add i to the set $\mathcal{I}$.

Proposition 3: Construction 5 gives a code over $GF(q)$ with r redundant symbols that can correct a single error in $\mathcal{E}$.

Proof: Let H be of size $r \times N$. Consider a single-error vector of length N that is either all zeros or contains a single non-zero element from $\mathcal{E}$. Note that the syndrome equals to the product of H and the single-error vector. We will show that the syndromes of such error vectors are distinct. Then the decoder can correct the error from the syndrome.

It is apparent that only when no error occurs, the syndrome is all zeros. So we consider the syndrome of two distinct single-error vectors, which can be written as $e_1\mathbf{h}'_1, e_2\mathbf{h}'_2$. Here, e_1, e_2 are elements of $\mathcal{E}$ representing the error values, and $\mathbf{h}'_1, \mathbf{h}'_2$ are columns of the parity check matrix H representing the error locations. Noticing that each column of H can be viewed as a major column multiplied by some non-zero scalar

TABLE V: Remainder error patterns, for $l = 1$. The primitive polynomial of $GF(27)$ is chosen to be $x^3 + 2x + 1$. The 3 coefficients in the polynomial representation correspond to the first 3 entries of the remainder tuple. Every element in $GF(27)$ is represented by an integer between 0 and 26, and in particular, each non-zero element is denoted by the integer exponent in the power representation plus 1.

Remainders	Power	Polynomial	Integer
0,0,1,2	1	1	1
0,1,0,2	α	α	2
2,0,0,1	α^2	α^2	3
0,1,2,0	α^3	$\alpha + 2$	4
1,2,0,0	α^4	$\alpha^2 + 2\alpha$	5
1,0,2,0	α^{12}	$\alpha^2 + 2$	13
0,0,2,1	α^{13}	2	14
0,2,0,1	α^{14}	2α	15
2,0,0,1	α^{15}	α^2	16
0,2,1,0	α^{16}	$2\alpha + 1$	17
2,1,0,0	α^{17}	$2\alpha^2 + \alpha$	18
2,0,1,0	α^{25}	$2\alpha^2 + 1$	26

from $GF(q)$, we can rewrite $\mathbf{h}'_1 = i_1\mathbf{h}_1, \mathbf{h}'_2 = i_2\mathbf{h}_2$, where $\mathbf{h}_1, \mathbf{h}_2$ are major columns, and $i_1, i_2 \in \mathcal{I}$ are the scalar multipliers. From (46), we know

$$i_1e_1 \neq i_2e_2. \tag{47}$$

When $\mathbf{h}_1 \neq \mathbf{h}_2$, since any two major columns in H are not linearly dependent, we know $i_1e_1\mathbf{h}_1 \neq i_2e_2\mathbf{h}_2$, or the syndromes are different.

When $\mathbf{h}_1 = \mathbf{h}_2$, due to (47), we see $i_1e_1\mathbf{h}_1 \neq i_2e_2\mathbf{h}_2$. ■

Now we apply Construction 5 to remainder classes with $(l = 1, t = 1)$ LMPE. From Table V, it is seen that the set of remainder error patterns is $\mathcal{E} = \{1, 2, 3, 4, 5, 13, 14, 15, 16, 18, 26\}$. By the condition in (46), H contains columns whose major elements are $1, 7$.

For example, the systematic parity check matrix for $r = 2$ is shown as below.

$$\begin{bmatrix} 0 & 1 & \dots & 1 & 7 & \dots & 7 & 1 & 0 \\ 7 & 1 & \dots & 26 & 0 & \dots & 26 & 0 & 1 \end{bmatrix}, \tag{48}$$

In general, the codeword length is $N = 2\frac{q^r-1}{q-1} = \frac{1}{13}(27^r - 1)$, for redundancy r.

Appendix D
Redundancy for the constructions

We derive the redundancy for general (l, t) in Proposition 4, and for $(l = 1, t = 1)$ in Proposition 5. They are summarized in Table IV. For simplicity, we assume that the required finite field size, e.g., the number of classes, is a prime power.

Proposition 4: The redundancy in bits for Constructions 2 and 3 approaches $2t\log_2(N + 1)$, and $3t\log_2(N + 1)$, respectively, when k/l is sufficiently large.

Proof: Similar to Figure 2, the parity symbols in the codeword contains the parity remainder part and the information quotient part. And the number of messages to be represented by the quotient part in one symbol depends on the parity remainders in the worst-case. One can show that the worst case occurs when the remainders sums to $4l+2+(k \mod (2l+1))$ or $6l + 3 + (k \mod (2l + 1))$, depending on whether the latter exceeds $8l$. And the corresponding number of possible information quotient tuples is C_{s+3}^3, where $s = \left\lfloor \frac{k}{2l+1} \right\rfloor - 2$ or $\left\lfloor \frac{k}{2l+1} \right\rfloor - 3$.

Redundancy for Construction 2. BCH code has length $q^m - 1 = N$, for the field size $q = (2l+1)^3$. The number of redundant symbols is no more than $r = 2mt = 2t \log_q(N+1)$. The total number of possible words is $(C_{k+3}^3)^N$, the number of information messages is $(C_{k+3}^3)^{N-r}(C_{s+3}^3)^r$, and the number of redundant bits is

$$\log_2(C_{k+3}^3)^N - \log_2(C_{k+3}^3)^{N-r}(C_{s+3}^3)^r \tag{49}$$

$$= \log_2 \frac{(C_{k+3}^3)^r}{(C_{s+3}^3)^r} \tag{50}$$

$$\approx r \log_2(2l+1)^3 \tag{51}$$

$$= 2t \log_2(N+1). \tag{52}$$

Here, (51) follows when k/l is large.

Redundancy for Construction 3. BCH code for the first layer has length $q_1^{m_1} - 1 = N$, for the field size $q_1 = (2l+1)^2$. The number of redundant symbols is no more than $r_1 = 2m_1 t = 2t \log_{q_1}(N+1)$. Note that q_1 is smaller than the field size in Construction 2, and thus the redundancy is longer. BCH code in the second layer has length $q_2^{m_2} - 1 = N$, field size $q_2 = 2l+1$, and distance $t+1$. The number of redundant symbols is $r_2 = m_2 t = r_1$. A symbol contains 3 parts: the class index (coded by BCH code in the first layer), the first entry of the remainder tuple (coded by BCH code in the second layer), and the quotients (information). Since the redundancy length is the same for both BCH codes, the r_1 parity symbols contain parity remainders and information quotients, and the remaining $N - r_1$ symbols contain only information. See Figure 4 (a). Thus, the number of information messages is $(C_{k+3}^3)^{N-r_1}(C_{s+3}^3)^{r_1}$, and the number of redundant bits has the same expression as (51) except that r is different:

$$r_1 \log_2(2l+1)^3 \tag{53}$$

$$= 3t \log_2(N+1). \tag{54}$$

The proof is completed. ■

Remark 2: In Figure 2, the number of possible quotients in the parity symbols are only 10, which was to accommodate the worst-case. The proof of the above proposition (in particular, the approximation step in (51)) implies that when the resolution k is large, we can ignore the loss for the number of possible quotients in the parity symbols, and only consider the redundancy in terms of the parity remainders.

Remark 3: Since the number of redundant symbols for BCH codes in the two layers are the same in Construction 3, the code structure is similar to Construction 2. Namely, the codeword can be divided into information symbols and parity symbols, where only remainders in the parity symbols are redundant.

Below we derive the redundancy for naive Hamming code (applied to the entire symbols), Hamming code (applied to the remainder classes as in Construction 2), improved Hamming code (in Construction 4), and Hamming code with reduced classes (in Construction 3), which are shown in Table IV.

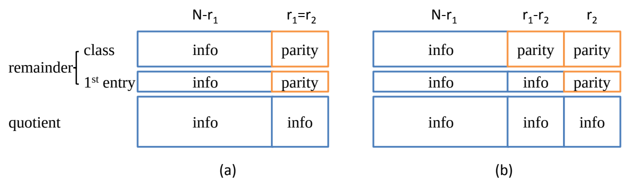

Fig. 4: Code structures for Construction 3 with reduced classes. (a) BCH code. (b) Hamming code.

Proposition 5: Let $l = 1, t = 1$. The redundancy in bits for naive Hamming code, Hamming code, improved Hamming code, and Hamming code with reduced classes is $\log_2\left(\left(C_{k+3}^3 - 1\right)N + 1\right)$, $\log_2(26N+1)$, $\log_2(13N+1)$, and $\log_2(8N+1) + \log_2(3)$, respectively, when k is sufficiently large.

Proof: Similar to the proof of Proposition 4, for the parity symbols, the number of possible information quotient tuples is C_{s+3}^3, where $s = \left\lfloor \frac{k}{2l+1} \right\rfloor - 2$.

Hamming code over $GF(q)$ has length $N = \frac{q^r - 1}{q-1}$, and the number of redundant symbols is $r = \log_q((q-1)N+1)$.

Redundancy for naive Hamming code. The naive Hamming code requires a field size $q = C_{k+3}^3$. The number of redundant symbols r can be represented as $r = \log_q\left(\left(C_{k+3}^3 - 1\right)N + 1\right)$. The total number of possible words is $(C_{k+3}^3)^N$, and the number of information messages is $(C_{k+3}^3)^{N-r}(C_{s+3}^3)^r$. Similar to (51), we can get the he number of redundant bits is $\log_2\left(\left(C_{k+3}^3 - 1\right)N + 1\right)$.

Redundancy for Hamming code. The remainder classes require a field size $q = 27$. The number of redundant symbols is $r = \log_q(26N+1)$. The number of information messages is $(C_{k+3}^3)^{N-r}(C_{s+3}^3)^r$. Similar to (51), we can get the he number of redundant bits is $\log_2(26N+1)$.

Redundancy for improved Hamming code. The improved Hamming code has length $N = \frac{1}{13}(27^r - 1)$, for the field size $q = 27$. The number of redundant symbols is $r = \log_q(13N+1)$. The number of information messages is $(C_{k+3}^3)^{N-r}(C_{s+3}^3)^r$. Similar to (51), we can get the he number of redundant bits is $\log_2(13N+1)$.

Redundancy for Hamming code with reduced classes. The Hamming code for the first layer has the field size $q_1 = 9$. The number of redundant symbols is than $r_1 = \log_{q_1}(8N+1)$. The code in the second layer has the field size $q_2 = 3$, and distance $t+1 = 2$, which is a single parity check code. The number of redundant symbols is $r_2 = \log_{q_2}(3) = 1$. A symbol contains 3 parts: the class index (9 possibilities) coded by the Hamming code in the first layer, the first entry of the remainder tuple (3 possibilities) coded by the single parity check code in the second layer, and the information quotients. The first $N - r_1$ symbols are information, the next $r_1 - r_2$ symbols contains parity class indices, and the last r_2 symbols contains parity remainders, shown in Figure 4 (b). The number of possible information messages is $(C_{k+3}^3)^{N-r_1}(C_{s+3}^3)^{r_1} 3^{r_1 - r_2}$. Similar to (51), we can approximate the the number of redundant bits for large k, which is

$$\begin{aligned}
&\log_2(C^3_{k+3})^N - \log_2((C^3_{k+3})^{N-r_1}(C^3_{s+3})^{r_1}3^{r_1-r_2})\\
\approx& r_1\log_{q_1}(2l+1)^3 - (r_1-r_2)\log_2(3)\\
=&\log_2(8N+1)+\log_2(3).
\end{aligned}$$

The proof is completed. ■

References

[1] J. Cox, "Long-term data storage in DNA," *Trends in biotechnology*, vol. 19, pp. 247–50, 08 2001.

[2] V. Zhirnov, R. Zadegan, G. Sandhu, G. Church, and W. Hughes, "Nucleic acid memory," *Nature Materials*, vol. 15, pp. 366–370, 03 2016.

[3] S. M. H. Tabatabaei Yazdi, Y. Yuan, J. Ma, H. Zhao, and O. Milenkovic, "A rewritable, random-access DNA-based storage system," *Scientific Reports*, vol. 5, 05 2015.

[4] H. Lee, R. Kalhor, N. Goela, J. Bolot, and G. Church, "Terminator-free template-independent enzymatic DNA synthesis for digital information storage," *Nature Communications*, vol. 10, 06 2019.

[5] G. Church, Y. Gao, and S. Kosuri, "Next-generation digital information storage in DNA," *Science (New York, N.Y.)*, vol. 337, p. 1628, 08 2012.

[6] N. Goldman, P. Bertone, S. Chen, C. Dessimoz, E. Leproust, B. Sipos, and E. Birney, "Towards practical, high-capacity, low-maintenance information storage in synthesized DNA," *Nature*, vol. 494, 01 2013.

[7] Y. Erlich and D. Zielinski, "DNA fountain enables a robust and efficient storage architecture," *Science*, vol. 355, pp. 950–954, 03 2017.

[8] L. Anavy, I. Vaknin, O. Atar, R. Amit, and Z. Yakhini, "Data storage in DNA with fewer synthesis cycles using composite DNA letters," *Nature Biotechnology*, vol. 37, 10 2019.

[9] S. Al-Bassam and B. Bose, "Asymmetric/unidirectional error correcting and detecting codes," *IEEE Transactions on Computers*, vol. 43, no. 5, pp. 590–597, 1994.

[10] M. Blaum and H. Van Tilborg, "On t-error correcting/all unidirectional error detecting codes," *IEEE Transactions on Computers*, vol. 38, no. 11, pp. 1493–1501, 1989.

[11] Bose and D. J. Lin, "Systematic unidirectional error-detecting codes," *IEEE Transactions on Computers*, vol. C-34, no. 11, pp. 1026–1032, 1985.

[12] R. Ahlswede, H. Aydinian, and L. Khachatrian, "Unidirectional error control codes and related combinatorial problems," 01 2002.

[13] Y. Cassuto, M. Schwartz, V. Bohossian, and J. Bruck, "Codes for asymmetric limited-magnitude errors with application to multilevel flash memories," *IEEE Transactions on Information Theory*, vol. 56, no. 4, pp. 1582–1595, 2010.

[14] N. Elarief and B. Bose, "Limited magnitude error detecting codes over Zq," in *2009 Information Theory and Applications Workshop*, 2009, pp. 29–33.

[15] ——, "Optimal, systematic, q-ary codes correcting all asymmetric and symmetric errors of limited magnitude," *IEEE Transactions on Information Theory*, vol. 56, no. 3, pp. 979–983, 2010.

[16] R. Gabrys, H. M. Kiah, and O. Milenkovic, "Asymmetric Lee distance codes for DNA-based storage," *IEEE Transactions on Information Theory*, vol. 63, no. 8, pp. 4982–4995, 2017.

[17] S. B. Gashkov and I. S. Sergeev, "Complexity of computation in finite fields," *Journal of Mathematical Sciences*, vol. 191, no. 5, pp. 661–685, 2013.

[18] W. Zhang and Z. Wang, "Limited-magnitude error correction for probability vectors in DNA storage," 2021, https://faculty.sites.uci.edu/zhiying/publications/.

[19] R. W. Hamming, "Error detecting and error correcting codes," *The Bell System Technical Journal*, vol. 29, no. 2, pp. 147–160, 1950.

[20] S. Lin and D. J. Costello, *Error control coding, Second edition*. USA: Prentice-Hall, Inc., 2004.

[21] A. Das and N. A. Touba, "Efficient non-binary Hamming codes for limited magnitude errors in MLC PCMs," in *2018 IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFT)*, 2018, pp. 1–6.