Clustering-Correcting Codes

Tal Shinkar,* **Eitan Yaakobi**,* **Andreas Lenz**,† and **Antonia Wachter-Zeh**,†
*Department of Computer Science, Technion — Israel Institute of Technology, Haifa, 3200009 Israel
†Institute for Communications Engineering, Technical University of Munich, Munich 80333, Germany
{tal.s,yaakobi}@cs.technion.ac.il, andreas.lenz@mytum.de, antonia.wachter-zeh@tum.de

***Abstract*—A new family of codes, called *clustering-correcting codes*, is presented in this paper. This family of codes is motivated by the special structure of data that is stored in DNA-based storage systems. The data stored in these systems has the form of unordered sequences, also called *strands*, and every strand is synthesized thousands to millions of times, where some of these copies are read back during sequencing. Due to the unordered structure of the strands, an important task in the decoding process is to place them in their correct order. This is usually accomplished by allocating a part of the strand for an index. However, in the presence of errors in the index field, important information on the order of the strands may be lost.**

Clustering-correcting codes ensure that if the distance between the index fields of two strands is small, then there will be a large distance between their data fields. It is shown how this property enables to place the strands together in their correct clusters even in the presence of errors. We present lower and upper bounds on the size of clustering-correcting codes and an explicit construction of these codes which uses only a single bit of redundancy.

I. Introduction

Progress in synthesis and sequencing technologies have paved the way for the development of a non-volatile data storage based on DNA molecules. The first large-scale experiments that demonstrated the potential of in vitro DNA storage were reported by Church et al. who recovered 643 KB of data [4] and Goldman et al. who accomplished the same task for a 739 KB message [8]. However, in both of these works the data was not recovered successfully due to the lack of using the appropriate coding solutions to correct errors. Since then, several more groups have demonstrated the ability to successfully store data of large scale using DNA molecules; see e.g. [1], [2], [5], [13], [18]. Other works developed coding solutions which are specifically targeted to correct the special types of errors inside DNA-based storage systems [10]–[12], [14], [16]–[18].

A DNA storage system consists of three steps. First, the strands containing the encoded data are synthesized. They are then stored inside a storage container, and finally a DNA sequencer reads back the strands. The encoding and decoding are two external processes to the system that convert the data to DNA strands and back. The structure of a DNA storage system is different from all other existing storage systems. Since the strands are stored unordered, it is unclear what part of the data each strand represents, even if no error occurred. For more details we refer the reader to [9], [11] and referencers therein.

Storing DNA strands in a way that will allow to reconstruct them back in the right order is an important task. The common solution to address this problem is to use indices, that are stored as part of the strand. Each strand is prefixed with some nucleotides that indicate the strand's location, with respect to all other strands. Although using indices is a simple solution it has several drawbacks. One of them is that in case of an error within the index, important information on the strand's location is lost as well as the ability to place it in the correct position between the other strands. In this paper a new coding scheme, called *clustering-correcting codes*, is presented which enables to combat errors with minimal redundancy.

In DNA storage systems, every strand is synthesized thousands of times (or even millions) and thus more than a single copy of each strand is read back upon sequencing. Thus, the first task based upon the sequencer's input is to partition all the reads into clusters such that all read strands at each cluster are copies of the same information strand. A possible solution is to use the indices in order to identify the strands and cluster them together, but in the presence of errors, this may result with mis-clustered strands which can cause errors in the recovered data. Hence, finding codes and algorithms for the clustering process is an important challenge. A naive solution is to add redundancy to the index part in order to correct potential errors in the index [3]. However, this will incur an unavoidable reduction in the storage rate of the DNA storage system. We will show in this paper how clustering-correcting codes can enable one to cluster all strands in the right clusters even with the presence of errors in the indices (see Fig. 1), while the redundancy is minimized. In fact, for a large range of parameters this can be done with only a single bit of redundancy for all the strands together.

The rest of the paper is organized as follows. In Section II, the family of clustering-correcting codes is presented as well as other useful definitions that will be used throughout the paper. In Section III, we present explicit and asymptotic lower and upper bounds on the size of clustering-correcting codes. In Section IV, we present an explicit construction of these codes which uses only a single bit of redundancy. Due to the lack of space, some of the proofs are omitted in the paper.

II. Definitions and Preliminaries

For a positive integer n, the set $\{0, 1, \dots, n-1\}$ is denoted by $[n]$. For two vectors $\boldsymbol{x}, \boldsymbol{y} \in \{0,1\}^n$ of the same length, we denote the i-th symbol of $\boldsymbol{x}$ by x_i. The subvector of $\boldsymbol{x}$ starting at the i-th index of length ℓ is denoted by $\boldsymbol{x}_{[i,\ell]}$. The Hamming distance between $\boldsymbol{x}$ and $\boldsymbol{y}$ is denoted by $d_H(\boldsymbol{x}, \boldsymbol{y})$ and the Hamming weight of $\boldsymbol{x}$ is $w_H(\boldsymbol{x})$. The radius-r ball of a vector $\boldsymbol{x} \in \{0,1\}^n$ is $B_r(\boldsymbol{x}) = \{\boldsymbol{y} \mid d_H(\boldsymbol{x}, \boldsymbol{y}) \leqslant r\}$ and its size is denoted by $B_n(r) \triangleq \sum_{i=0}^{r} \binom{n}{i}$. The function $\mathcal{H}(x)$ for $0 \leqslant x \leqslant 1$ denotes the binary entropy function, and the inverse function $\mathcal{H}^{-1}(x)$ for $0 \leqslant x \leqslant 1$ is defined to return values between 0 and 1/2. We study here the binary case while extensions to the non-binary case are straightforward.

Assume M strands are stored in a DNA-based storage system where the size of every strand is L. We will assume that $M = 2^{\beta L}$ for some $0 < \beta < 1$ and for simplicity, it is assumed that M is a power of 2. For any integer $i \in [M]$, its binary representation of length $\log(M)$ is denoted by ind_i. Every length-L strand s that will be stored in the system is of the form $s = (\mathsf{ind}, \boldsymbol{u})$, where ind is the length-$\log(M)$ *index field* of the strand (the binary representation of a number between 0 and $M-1$) and $\boldsymbol{u}$ is the *data field* of $L - \log(M)$ bits that are used

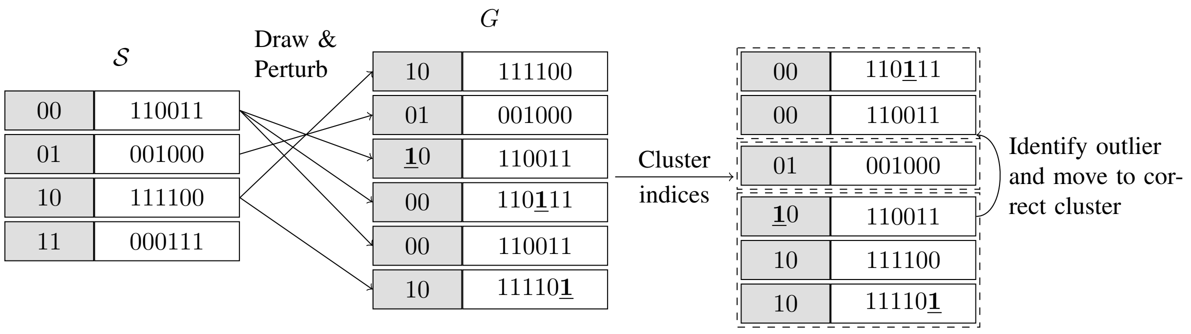

Fig. 1: Exemplary realization of the DNA channel model. A set $\mathcal{S}$ of $M = 4$ strands is stored and $N = 6$ strands are drawn with errors (highlighted in bold). The strands are clustered according to their indices. The outlier can be identified as it has large distance w.r.t. all other strands in the cluster and be put into the correct cluster.

to store the information or the redundancy of an error-correcting code. Every stored message will have M strands of this form and the space of all possible messages that can be stored in the DNA storage system is

$$\mathcal{X}_{M,L}=\{\{(\mathsf{ind}_0,\boldsymbol{u}_0),\ldots,(\mathsf{ind}_{M-1},\boldsymbol{u}_{M-1})\}|\boldsymbol{u}_j\in\{0,1\}^{L-\log(M)}\}.$$

Clearly, $|\mathcal{X}_{M,L}| = 2^{M(L-\log(M))}$. Under this setup, a code $\mathcal{C}$ will be a subset of $\mathcal{X}_{M,L}$, where each codeword $\mathcal{S}$ of $\mathcal{C}$ is a subset of the form $\mathcal{S} = \{(\mathsf{ind}_0,\boldsymbol{u}_0),\ldots,(\mathsf{ind}_{M-1},\boldsymbol{u}_{M-1})\}$. For shorthand, the term $L-\log(M)$ will be abbreviated by L_M.

When a set $\mathcal{S} = \{(\mathsf{ind}_0,\boldsymbol{u}_0),\ldots,(\mathsf{ind}_{M-1},\boldsymbol{u}_{M-1})\}$ is synthesized, each of its strands $(\mathsf{ind}_i,\boldsymbol{u}_i)$, which are called the *input strands*, has thousands to millions of copies and during the sequencing process a subset of these copies is read. Hence, the sequencer's output is another set G of some N strands, called the *output strands*, where N is significantly larger than M. Each output strand in the set G is a copy of one of the input strands in $\mathcal{S}$, however with some potential errors. A DNA-based storage system is called a *(τ,ρ)-DNA system* if it satisfies the following property: If the output strand $(\mathsf{ind}',\boldsymbol{u}')\in G$ is a noisy copy of the input strand $(\mathsf{ind},\boldsymbol{u})\in S$, then $d_H(\mathsf{ind},\mathsf{ind}')\leqslant\tau$ and $d_H(\boldsymbol{u},\boldsymbol{u}')\leqslant\rho$. That is, the index field has at most τ Hamming errors while the data field has at most ρ Hamming errors. We consider in this work only substitution errors while extensions for deletions, insertions, and more generally the edit distance will be analyzed in the full version of the paper.

Since the set G contains several noisy copies of each input strand in $\mathcal{S}$, the first task in the decoding process is to partition the set of all N output strands into M cluster sets, such that the output strands in every cluster are noisy copies of the same input strand. Since every strand contains an index, the simplest way to operate this task is by partitioning the output strands into M sets based upon the index field in every output strand. This process will indeed be successful if there are no errors in the index field of every output strand, however other solutions are necessary since the error rates in DNA storage systems are not negligible [9]. Another approach to cluster the strands is based upon the distances between every pair of output strands, as was studied in [14]. However, this approach suffers extremely high computational complexity.

In this work, we take a hybrid solution of these two approaches. First, we cluster the output strands based on the indices in the output strands. Then, we scan for output strands which were mis-clustered, that is, were placed in the wrong cluster. This is accomplished by checking the distances between the output strands in every cluster in order to either remove output strands that were incorrectly placed in a cluster due to errors in their index or move them to their correct cluster set. Since we compute the distances only between pairs of strands that were placed in the same cluster (and not between all pairs of strands), this step will result in a significantly lower complexity compared to the solution from [14]. However, in order to succeed in this new approach we need the strands stored in the set $\mathcal{S}$ to satisfy several constraints. These constraints will be met by the family of *clustering-correcting codes* which are presented in this paper. Another assumption taken in this model, which will be referred to as the *majority assumption*, assumes that in every cluster the majority of the strands have the correct index. Since the number of strands is very large this assumption holds with high probably if not in all cases.

The main idea to move strands which were misplaced in a cluster due to errors in their index field works as follows. Assume the strand $s_i = (\mathsf{ind}_i,\boldsymbol{u}_i)$ has a noisy copy of the form $s_i' = (\mathsf{ind}_i',\boldsymbol{u}_i')$, and let j be such that $s_j = (\mathsf{ind}_j,\boldsymbol{u}_j)$ and $\mathsf{ind}_j = \mathsf{ind}_i'$. We need to make sure that the distance between $\boldsymbol{u}_i'$ and $\boldsymbol{u}_j$ is large enough as this will allow to identify that the output strand s_i' is erroneous and therefore does not belong to the cluster of index ind_j; see Fig. 1. We will be interested in either identifying that the output strand s_i' does not belong to this cluster or more than that, place it in its correct cluster. This motivates us to study the following family of codes.

Definition 1. *A word $\mathcal{S} = \{(\mathsf{ind}_0,\boldsymbol{u}_0),\ldots,(\mathsf{ind}_{M-1},\boldsymbol{u}_{M-1})\}\in\mathcal{X}_{M,L}$ is said to satisfy the (e,t)-**clustering constraint** if for all $(\mathsf{ind}_i,\boldsymbol{u}_i),(\mathsf{ind}_j,\boldsymbol{u}_j)\in\mathcal{S}$ in which $i\neq j$ and $d_H(\mathsf{ind}_i,\mathsf{ind}_j)\leqslant e$, it holds that $d_H(\boldsymbol{u}_i,\boldsymbol{u}_j)\geqslant t$.*

*A code $\mathcal{C}\subseteq\mathcal{X}_{M,L}$ will be called an (e,t)-**clustering-correcting code (CCC)** if every $\mathcal{S}\in\mathcal{C}$ satisfies the (e,t)-clustering constraint.*

The *redundancy* of a code $\mathcal{C}\subseteq\mathcal{X}_{M,L}$ will be defined by

$$r = ML_M - \log|\mathcal{C}|.$$

Our goal in this work is to find (e,t)-CCCs for all e and t. We denote by $A_{M,L}(e,t)$ the size of the largest (e,t)-CCC in $\mathcal{X}_{M,L}$, and by $r_{M,L}(e,t)$ the optimal redundancy of an (e,t)-CCC, so $r_{M,L}(e,t) = ML_M - \log(A_{M,L}(e,t))$.

The clustering-correcting capabilities of CCCs are proved in the next theorem. We note that as a result of this theorem, if the number of errors is not too large, it is already possible to place every output strand in its correct cluster.

Theorem 2. *For fixed integers M, L, e, t, let $\mathcal{C}$ be an (e,t)-CCC. Assume that a set $\mathcal{S} \in \mathcal{C}$ is stored in a (τ,ρ)-DNA system. The following properties hold:*

1) *If $\tau \leqslant e$ and $4\rho < t$ then every output strand can be detected to be placed in a wrong cluster.*
2) *If $\tau \leqslant e/2$ and $4\rho < t$ then every output strand can be placed in its correct cluster.*

Proof: We prove only the first statement while the proof of second one is similar. Let $(\mathsf{ind}'_i, \boldsymbol{u}'_i)$ be a noisy copy of the strand $(\mathsf{ind}_i, \boldsymbol{u}_i)$. Since the data is stored in a (τ,ρ)-DNA system, it holds that $d_H(\mathsf{ind}_i, \mathsf{ind}'_i) \leqslant \tau$, and therefore $d_H(\mathsf{ind}_i, \mathsf{ind}'_i) \leqslant e$. Also $d_H(\boldsymbol{u}_i, \boldsymbol{u}'_i) \leqslant \rho$. Let $j \in [M]$ be such that $\mathsf{ind}_j = \mathsf{ind}'_i$. From the fact that $S \in \mathcal{C}$ we derive that $d_H(\boldsymbol{u}_i, \boldsymbol{u}_j) \geqslant t > 4\rho$, and thus $d_H(\boldsymbol{u}_j, \boldsymbol{u}'_i) \geqslant d_H(\boldsymbol{u}_i, \boldsymbol{u}_j) - d_H(\boldsymbol{u}_i, \boldsymbol{u}'_i) > 3\rho$. Let $(\mathsf{ind}_j, \boldsymbol{u}'_j)$ be a noisy copy strand of $(\mathsf{ind}_j, \boldsymbol{u}_j)$, that is, errors might occur in the data field but not in the index field. So, $d_H(\boldsymbol{u}_j, \boldsymbol{u}'_j) \leqslant \rho$ which yields that $d_H(\boldsymbol{u}'_i, \boldsymbol{u}'_j) > 2\rho$. On the other hand, the distance between the data fields of the two strands that belong to the same cluster is at most 2ρ. That is, under the majority assumption, a mis-clustered strand will have a distance of more than 2ρ from the majority of the strands in the cluster, and so it can be dropped instead of being mis-clustered. ■

The clustering algorithm according to Theorem 2 first partitions the strands to clusters according to the indices and then compares the distances between *only* strands in the same cluster. Hence, the number of comparisons is significantly smaller than the solution in [14], which compares between *all* strands. A rigorous analysis of the algorithm's complexity and its comparison with [14] will appear in the long version of the paper.

III. Bounds

Upper and lower bounds on $A_{M,L}(e,t)$ are presented. Let $D_n(d)$ be the size of the largest length-n error-correcting code $C \subseteq \{0,1\}^n$ and minimum Hamming distance d. For the rest of the paper, let $B_1 = B_{\log(M)}(e) - 1, B_2 = B_{L_M}(t-1)$, $\mathsf{le} = \log(\exp(1)) \approx 1.44$, and it is also assumed that $\beta < 1/2$.

Theorem 3. *For all M, L, e, and t it holds that*

$$A_{M,L}(e,t) \geqslant 2^{ML_M}\left(1 - \frac{B_1B_2}{2^{L_M}}\right)^{M-D}$$

and hence

$$r_{M,L}(e,t) \leqslant \frac{\mathsf{le}\cdot(M-D)B_1B_2}{2^{L_M} - B_1B_2},$$

where $D = D_{\log(M)}(e+1)$.

Proof: In order to verify the lower bound, we construct an (e,t)-CCC $\mathcal{C}$ that will yield a lower bound on $A_{M,L}(e,t)$. Let $C_1 \subseteq \{0,1\}^{\log(M)}$ be a length-$\log(M)$ code with minimum Hamming distance $e+1$ of size D. Each codeword $\mathcal{S} = \{(\mathsf{ind}_0, \boldsymbol{u}_0), \dots, (\mathsf{ind}_{M-1}, \boldsymbol{u}_{M-1})\} \in \mathcal{C}$ is constructed in two steps. First, we choose the data field of strands with indices that belong to the code C_1, that is, all strands of the form $(\mathsf{ind}, \boldsymbol{u})$ such that $\mathsf{ind} \in C_1$. There are 2^{L_M} options for each strand and thus $(2^{L_M})^D$ options for the first step. Since the Hamming distance between all pairs of indices of these strands is at least $e+1$, their data fields can be chosen independently.

For the rest of the strands we assume the worst case. That is, for each strand left, all of its neighbors are chosen and their radius-$(t-1)$ balls are mutually disjoint. Thus, there are at least

$$2^{L_M} - (B_{\log(M)}(e) - 1)\cdot B_{L_M}(t-1) = 2^{L_M} - B_1B_2$$

options to choose the data field of each remaining strand. In conclusion, there are $2^{L_M D}\left(2^{L_M} - B_1B_2\right)^{M-D}$ options for choosing a valid set $\mathcal{S} \in \mathcal{C}$, and hence

$$A_{M,L}(e,t) \geqslant 2^{ML_M}\left(1 - \frac{B_1B_2}{2^{L_M}}\right)^{M-D}.$$

We can also deduce an upper bound on the redundancy

$$r_{M,L}(e,t) \leqslant \frac{\mathsf{le}\cdot(M-D)B_1B_2}{2^{L_M} - B_1B_2},$$

where here the inequality $-\log(1-x) \leqslant \mathsf{le}\cdot\frac{x}{1-x}$ for all $0 < x < 1$ is used. ■

The next corollary follows directly from Theorem 3.

Corollary 4. *If $t \leqslant L_M\mathcal{H}^{-1}\left(\frac{1-2\beta}{1-\beta} - \frac{\log(\beta L)}{(1-\beta)L}\right)$ then $r_{M,L}(1,t) \leqslant 1$.*

Proof: For $e = 1$, $D_{\log(M)}(2) = M/2$. This is achieved by selecting all indices ind_i such that $w_H(\mathsf{ind}_i)$ is even (or odd). Hence, from Theorem 3 it holds that

$$r_{M,L}(1,t) \leqslant \frac{\mathsf{le}\cdot M\log(M)B_2}{2^{L_M+1} - 2\log(M)B_2}.$$

Hence, $r_{M,L}(1,t) \leqslant 1$ if $B_2 \leqslant \frac{2^{L_M+1}}{\log(M)(\mathsf{le}\cdot M+2)}$. According to Lemma 4.7 in [15], $B_2 \leqslant 2^{L_M\mathcal{H}\left(\frac{t-1}{L_M}\right)}$ and hence it is enough to require that $2^{L_M\mathcal{H}\left(\frac{t-1}{L_M}\right)} \leqslant \frac{2^{L_M+1}}{\log(M)(\mathsf{le}\cdot M+2)}$, i.e.,

$$\begin{aligned}L_M\mathcal{H}\left(\frac{t-1}{L_M}\right) &\leqslant L_M - \log(M) - \log\log(M)\\ &\leqslant L_M + 1 - \log(\mathsf{le}\cdot M + 2) - \log\log(M).\end{aligned}$$

For $M = 2^{\beta L}$, this holds for all $t \leqslant L_M\mathcal{H}^{-1}\left(\frac{1-2\beta}{1-\beta} - \frac{\log(\beta L)}{(1-\beta)L}\right)$. ■

A similar upper bound on $A_{M,L}(e,t)$ is presented next.

Theorem 5. *For all M, L, e and t it holds that*

$$A_{M,L}(e,t) \leqslant 2^{ML_M}\left(1 - \frac{B_2}{2^{L_M}}\right)^{M-1},$$

and therefore $r_{M,L}(e,t) \geqslant \frac{\mathsf{le}\cdot(M-1)\cdot B_2}{2^{L_M}}$. Furthermore, if $t \geqslant L_M\cdot\mathcal{H}^{-1}\left(\frac{1-2\beta}{1-\beta} + \frac{\log(L_M)}{L_M}\right) + 1$ then $r_{M,L}(1,t) \geqslant 1$.

From Corollary 4 and Theorem 5 we get that for L_M large enough and $t \approx L_M\cdot\mathcal{H}^{-1}\left(\frac{1-2\beta}{1-\beta}\right)$, we get that $r_{M,L}(1,t) \approx 1$. An asymptotic improvement to the upper bound from Theorem 5 for $e = 1$ which matches the lower bound from Theorem 3 is proved in the next theorem.

Theorem 6. *For L_M large enough, if $\frac{t}{L_M} < \frac{1}{2}\mathcal{H}^{-1}\left(\frac{1-2\beta}{1-\beta}\right) - \epsilon$, for some fixed $\epsilon > 0$, then it holds that*

$$A_{M,L}(1,t) \leqslant 2^{ML_M}\left(1 - \frac{\log(M)B_2}{2^{L_M}}\right)^{M/2}(1+\delta),$$

for δ small enough, and hence

$$r_{M,L}(1,t) \geqslant \frac{M\log(M)B_2}{2^{L_M+1} - 2\log(M)B_2} - \log(1+\delta).$$

Proof: Let $\mathcal{C}$ be a $(1,t)$-CCC of maximal size $A_{M,L}(1,t)$. For every set $\mathcal{S} = \{(\mathsf{ind}_0, \boldsymbol{u}_0), \dots, (\mathsf{ind}_{M-1}, \boldsymbol{u}_{M-1})\} \in \mathcal{C}$, let

$$\mathcal{S}_{\text{even}} = (\boldsymbol{u}_i)_{i: w_H(\mathsf{ind}_i)\text{ is even}} \in (\{0,1\}^{L_M})^{M/2} \triangleq \Sigma_{M,L}$$

be the *vector* projection of $\mathcal{S}$ to the data fields of the strands with indices of even weight and let $I_{\text{even}} \triangleq \{i \mid w_H(\mathsf{ind}_i) \text{ is even}\}$.

The sets of the strands in the code $\mathcal{C}$ are partitioned according to their projection on the indices with even weight. More specifically, for every $\boldsymbol{v} \in \Sigma_{M,L}$, let $\mathcal{C}_{\boldsymbol{v}}$ be the subcode of $\mathcal{C}$,

$$\mathcal{C}_{\boldsymbol{v}} = \{\mathcal{S} \in \mathcal{C} \mid \mathcal{S}_{\text{even}} = \boldsymbol{v}\},$$

so it holds that $\mathcal{C} = \bigcup_{\boldsymbol{v}\in\Sigma_{M,L}} \mathcal{C}_{\boldsymbol{v}}$.

A vector $\boldsymbol{v} = (\boldsymbol{v}_i)_{i\in I_{\text{even}}} \in \Sigma_{M,L}$ is *good* if for all $i, j \in I_{\text{even}}$ such that $d_H(\mathsf{ind}_i, \mathsf{ind}_j) = 2$ it holds that $B_{t-1}(\boldsymbol{v}_i) \cap B_{t-1}(\boldsymbol{v}_j) = \emptyset$, and otherwise it is *bad*. Denote by $X_{\text{good}}, X_{\text{bad}}$ the number of good, bad vectors in $\Sigma_{M,L}$, respectively. If a vector $\boldsymbol{v} \in \Sigma_{M,L}$ is bad, then there are at least two indices $i, j \in I_{\text{even}}$ such that $d_H(\mathsf{ind}_i, \mathsf{ind}_j) = 2$ and $B_{t-1}(\boldsymbol{v}_i) \cap B_{t-1}(\boldsymbol{v}_j) \neq \emptyset$, i.e., $d_H(\boldsymbol{v}_i, \boldsymbol{v}_j) \leqslant 2t-2$. Hence, we get that

$$X_{\text{bad}} \leqslant M(\log(M))^2 B_3 \cdot 2^{L_M(\frac{M}{2}-1)},$$

where $B_3 = B_{L_M}(2t-2)$.

Consider the size of $\mathcal{C}_{\boldsymbol{v}}$ when $\boldsymbol{v}$ is good. For every $\mathcal{S} \in \mathcal{C}_{\boldsymbol{v}}$, we only need to assign the data fields for strands of odd weight index. Since $\boldsymbol{v}$ is a good vector, for every index of odd weight, the radius-$(t-1)$ balls of all of its neighbor strands are mutually disjoint so there are exactly

$$2^{L_M} - \log(M) \cdot B_2$$

options to choose the data field of the i-th strand. For every bad vector $\boldsymbol{v} \in \Sigma_{M,L}$, it is enough to take the loose bound in which $|\mathcal{C}_{\boldsymbol{v}}| \leqslant \left(2^{L_M}\right)^{\frac{M}{2}}$. In conclusion we get that

$$\begin{aligned}
|\mathcal{C}| &= \Big|\bigcup_{\boldsymbol{v}\in\Sigma_{M,L}} \mathcal{C}_{\boldsymbol{v}}\Big| = \Big|\bigcup_{\boldsymbol{v}\in\Sigma_{M,L}:\boldsymbol{v}\text{ is good}} \mathcal{C}_{\boldsymbol{v}}\Big| + \Big|\bigcup_{\boldsymbol{v}\in\Sigma_{M,L}:\boldsymbol{v}\text{ is bad}} \mathcal{C}_{\boldsymbol{v}}\Big| \\
&\leqslant X_{\text{good}}\left(2^{L_M} - \log(M)B_2\right)^{\frac{M}{2}} + X_{\text{bad}}\left(2^{L_M}\right)^{\frac{M}{2}} \\
&\leqslant 2^{\frac{ML_M}{2}}\left(2^{L_M} - \log(M)B_2\right)^{\frac{M}{2}} + X_{\text{bad}} 2^{\frac{ML_M}{2}} \\
&\leqslant 2^{ML_M}\left(1 - \frac{\log(M)B_2}{2^{L_M}}\right)^{\frac{M}{2}} + \frac{M(\log(M))^2 B_3 2^{ML_M}}{2^{L_M}} \\
&= 2^{ML_M}\left(1 - \frac{\log(M)B_2}{2^{L_M}}\right)^{\frac{M}{2}}\left(1 + \frac{M(\log(M))^2 B_3}{2^{L_M}\cdot\left(1 - \frac{\log(M)\cdot B_2}{2^{L_M}}\right)^{\frac{M}{2}}}\right).
\end{aligned}$$

According to $-\log(1-x) \leqslant \mathsf{le} \cdot \frac{x}{1-x}$ for all $0 < x < 1$ we get

$$\left(1 - \frac{\log(M)B_2}{2^{L_M}}\right)^{\frac{M}{2}} \geqslant 2^{-\frac{\mathsf{le}\cdot\log(M)B_2(M/2)}{2^{L_M}-\log(M)B_2}}.$$

We use again the inequality $B_2 \leqslant 2^{L_M\mathcal{H}\left(\frac{t-1}{L_M}\right)}$ and $B_3 \leqslant 2^{L_M\mathcal{H}\left(\frac{2(t-1)}{L_M}\right)}$, while for $\frac{t}{L_M} < \frac{1}{2}\mathcal{H}^{-1}\left(\frac{1-2\beta}{1-\beta}\right) - \epsilon$ it holds

$$\begin{aligned}
&\lim_{L_M\to\infty} \frac{\mathsf{le}\cdot\log(M)B_2(M/2)}{2^{L_M} - \log(M)B_2} \leqslant \lim_{L_M\to\infty} \frac{\mathsf{le}\cdot\log(M)2^{L_M\mathcal{H}\left(\frac{t-1}{L_M}\right)}2^{\beta L-1}}{2^{L_M} - \log(M)2^{L_M\mathcal{H}\left(\frac{t-1}{L_M}\right)}} \\
&= \lim_{L_M\to\infty} \frac{\mathsf{le}\cdot\log(M)2^{L_M\mathcal{H}\left(\frac{t-1}{L_M}\right)}2^{\beta L-1}}{2^{L_M}} = 0,
\end{aligned}$$

and

$$\begin{aligned}
&\lim_{L_M\to\infty} \frac{M(\log(M))^2 B_3}{2^{L_M}} \leqslant \lim_{L_M\to\infty} \frac{M(\log(M))^2 2^{L_M\mathcal{H}\left(\frac{2(t-1)}{L_M}\right)}}{2^{L_M}} \\
&= \lim_{L\to\infty} \frac{2^{\beta L}(\beta L)^2 2^{(1-2\beta-\epsilon')L}}{2^{(1-\beta)L}} = \lim_{L\to\infty} \frac{(\beta L)^2}{2^{\epsilon' L}} = 0,
\end{aligned}$$

for some $\epsilon' > 0$. Thus, $\lim_{L\to\infty} \frac{ML^{2t}}{2^{L_M}\cdot\left(1 - \frac{\log(M)\cdot B_2}{2^{L_M}}\right)^{\frac{M}{2}}} = 0$, which confirms the theorem's statements.

■

IV. A Construction of CCCs

In this section we propose a construction of CCCs. It is shown that with a single bit of redundancy it is possible to construct CCCs for relatively large values of t.

The algorithm will use the following functions:

- The function $w_\ell(S, t)$ is defined over a set of vectors S and a positive integer t and outputs a vector $\boldsymbol{w} \in \{0,1\}^\ell$ which satisfies the following condition. For all $\boldsymbol{v} \in S$, $d_H(\boldsymbol{w}, \boldsymbol{v}_{[\log(M),\ell]}) \geqslant t$. The value of ℓ will be determined later as a function of e, t, and M.
- The function $\Delta_1(\mathsf{ind}_i, \mathsf{ind}_j)$ encodes the difference between the two indices i and j of Hamming distance at most e using $e\lceil\log(\log(M))\rceil$ bits which mark the positions where the indices $\mathsf{ind}_i, \mathsf{ind}_j$ differ.
- The function $\Delta_2(\boldsymbol{u}_i, \boldsymbol{u}_j)$ encodes the difference between the two data fields $\boldsymbol{u}_i, \boldsymbol{u}_j \in \{0,1\}^{L_M}$ of Hamming distance at most $t-1$ using $(t-1)\log(L_M)$ bits which mark the positions where they differ.

The input to the algorithm is M vectors $\boldsymbol{v}_0, \ldots, \boldsymbol{v}_{M-1}$. All vectors are of length L_M bits, except for $\boldsymbol{v}_{M-1}$ which has length of $L_M - 1$ bits. The idea behind the presented algorithm is to find all pairs of vectors that do not satisfy the clustering constraint, and correct them in a way that they satisfy the constraint and yet the original data can be uniquely recovered. A bit is added to $\boldsymbol{v}_{M-1}$, hence, the code has a single bit of redundancy, to mark whether some vectors were altered by the algorithm.

For $i \in [M]$, the notation $S(e, i)$ in the algorithm will be used as a shortcut to the set $\{\boldsymbol{u}_j \mid d_H(\mathsf{ind}_i, \mathsf{ind}_j) \leqslant e\}$ of data fields corresponding to indices ind_j of Hamming distance at most e from ind_i. At any iteration of the while loop, when the i-th strand is corrected, the function $w_\ell(S(e,i), t)$ will be used to update the data field $\boldsymbol{u}_i$ such that it does not violate the constraint and yet can be decoded. In order to make room for the vector generated by the function $w_\ell(S(e,i), t)$, we will encode $\boldsymbol{u}_i$ based on its similarity to one of its close neighbors $\boldsymbol{u}_j$. These modifications are encoded together as a repelling vector of length $len = \ell + \log\log(M) + (t-1)\log(L_M)$, when $e = 1$.

Algorithm 1 $(1, t) - CCC$ Construction

```
Input: $M$ vectors $\boldsymbol{v}_0, \ldots, \boldsymbol{v}_{M-1}$ such that $\boldsymbol{v}_0, \ldots, \boldsymbol{v}_{M-2} \in \{0,1\}^{L_M}$ and $\boldsymbol{v}_{M-1} \in \{0,1\}^{L_M-1}$
Output: a codeword $\mathcal{S} = \{(\mathsf{ind}_0, \boldsymbol{u}_0), \ldots, (\mathsf{ind}_{M-1}, \boldsymbol{u}_{M-1})\}$
1: $\forall i \in \{0, \ldots, M-2\} : \boldsymbol{u}_i = \boldsymbol{v}_i, \boldsymbol{u}_{M-1} = (\boldsymbol{v}_{M-1}, 0)$
2: $p = M - 1$
3: $B = \{(i,j) \mid i < j, d_H(\mathsf{ind}_i, \mathsf{ind}_j) \leqslant 1 \wedge d_H(\boldsymbol{u}_i, \boldsymbol{u}_j) < t\}$
4: while $B \neq \emptyset$, let $(i,j) \in B$ do
5:     $(\boldsymbol{u}_p)_{L_M-1} = 1$
6:     $(\boldsymbol{u}_p)_{[0,\log(M)]} = \mathsf{ind}_i$
7:     $p = i$
8:     $repl = (w_\ell(S(1,i), t), \Delta_1(\mathsf{ind}_i, \mathsf{ind}_j), \Delta_2(\boldsymbol{u}_i, \boldsymbol{u}_j))$
9:     $(\boldsymbol{u}_i)_{[\log(M),len]} = repl$
10:    $B = \{(i,j) \mid (i,j) \in B \wedge d_H(\boldsymbol{u}_i, \boldsymbol{u}_j) < t\}$
11: end while
12: $(\boldsymbol{u}_p)_{L_M-1} = 0$
13: $(\boldsymbol{u}_p)_{[0,\log(M)]} = (\boldsymbol{v}_{M-1})_{[0,\log(M)]}$
```

Theorem 7. *For any input vectors $\boldsymbol{v}_0, \ldots, \boldsymbol{v}_{M-1}$, Algorithm 1 returns a valid $(1,t)$-CCC codeword for any t that satisfies:*

$$\ell + \log\log(M) + (t-1)\log(L_M) + 1 \leqslant L_M - \log(M).$$

Furthermore, it is possible to decode the vectors $\boldsymbol{v}_0, \ldots, \boldsymbol{v}_{M-1}$, and Algorithm 1 uses a single bit of redundancy.

Proof: After the initializing steps, in Step 3 the algorithm gathers the indices of all pairs of strands that both their index and data fields are too close to each other, hence, violating the constraint. The algorithm iterates over the set B, handling one pair at a time. In Step 10, this set is updated and the algorithm stops when the set B is empty, i.e., there are no bad pairs and so $\{(\mathsf{ind}_0, \boldsymbol{u}_0), \ldots, (\mathsf{ind}_{M-1}, \boldsymbol{u}_{M-1})\}$ satisfies the constraint.

On each iteration of the while loop, the algorithm takes a pair of strands, say of indices i and j, where $i < j$, which violates the constraint and changes the data field in the i-th strand. First, the flag at the end of the previous strand is changed to 1 (Step 5). This denotes that it is not the last strand in the decoding chain. In Step 8, the algorithm calculates the vector $\boldsymbol{w} = w_\ell(S(1,i), t)$ and embeds it in the data field of the i-th strand. The vector $\boldsymbol{w}$ satisfies that for all $\boldsymbol{u} \in S(1,i)$, $d_H(\boldsymbol{w}, \boldsymbol{u}_{[\log(M),\ell]}) \geqslant t$. Then, for all $\boldsymbol{u} \in S(1,i)$ we have that $d_H((\boldsymbol{u}_i)_{[\log(M),\ell]}, \boldsymbol{u}_{[\log(M),\ell]}) \geqslant t$ and lastly for all $\boldsymbol{u} \in S(1,i)$, $d_H(\boldsymbol{u}_i, \boldsymbol{u}) \geqslant t$. Therefore the i-th and the j-th strands satisfy the constraint and thus do not belong to the set B when it is updated in Step 10. In fact any bad pair of indices which includes the i-th strand will be removed as well from the set B. Furthermore, since the i-th strand has been updated in such a way that it satisfies the constraint with respect to all of its neighbors, and the p-th strand in this iteration already satisfies the constraint according to his last $L_M - \log(M)$ bits, no bad pairs have been created. That is, the size of the set B decreases in each iteration, and the algorithm terminates. The constraint on t guarantees that the data field is large enough in order to write the information required on each update step of the while loop.

The idea of the decoding process is to track the updates chain of the strands and then traverse the chain in the opposite direction while recovering the input vectors. ■

Algorithm 1 can easily be extended to support larger values of e. In this case the constraint on t is

$$L - 2\log(M) \geqslant \ell + \log(B_{\log(M)}(e) \cdot B_{L_M}(t-1)) + 1.$$

Next we discuss the function $w_\ell(S,t)$. This function takes a set of vectors S as input, and outputs a vector $\boldsymbol{w} \in \{0,1\}^\ell$ such that for all $\boldsymbol{v} \in S_{[\log(M),\ell]} \triangleq \{\boldsymbol{v}_{[\log(M),\ell]} | \boldsymbol{v} \in S\}$, it holds that $d_H(\boldsymbol{w}, \boldsymbol{v}) \geqslant t$. The length ℓ of the vector $\boldsymbol{w}$ is determined by the minimum value of ℓ for which

$$2^\ell > |S_{[\log(M),\ell]}| \cdot B_\ell(t-1).$$

That is, the minimum length that allows us to choose a vector that does not fall into any of the radius-$(t-1)$ balls of the vectors in the set $S_{[\log(M),\ell]}$. The size of $S_{[\log(M),\ell]}$ is at most $B_{\log(M)}(e) - 1$, and we denote by $\ell(e,t,M)$ the smallest value of ℓ such that $2^\ell > (B_{\log(M)}(e) - 1) \cdot B_\ell(t-1)$.

Lemma 8. *For all e, t, M such that $t \leqslant \frac{(\log(M))^e}{e \cdot \log\log(M)}$, it holds that*

$$\ell(e,t,M) \leqslant et \cdot \log\log(M).$$

Corollary 9. *For all $t \leqslant \frac{L_M - \log(M) - e\log\log(M) + \log(L_M)}{\log(L_M) + e\log\log(M)}$ there exists an explicit construction of an (e,t)-CCC using Algorithm 1 which uses a single bit of redundancy.*

Proof: From Lemma 8 we can use $w_\ell(S,t)$ in Algorithm 1 with $\ell = et \cdot \log\log(M)$. In addition, from Theorem 7 the value of t should satisfy

$$\ell + \log(B_{\log(M)}(e) \cdot B_{L_M}(t-1)) + 1 \leqslant L_M - \log(M).$$

Thus, it is enough to show that

$$et \cdot \log\log(M) + \log((\log(M))^e \cdot {L_M}^{t-1}) \leqslant L_M - \log(M),$$

while using $B_{\log(M)}(e) \leqslant (\log(M))^e$ and $B_{L_M}(t-1) \leqslant {L_M}^{t-1}$. Hence, it is enough for t to satisfy

$$t(\log(L_M) + e\log\log(M)) \leqslant L_M - \log(M) - e\log\log(M) + \log(L_M).$$

■

According to Section III, $r_{M,L}(1,t) = 1$ when t is approximately $L_M \cdot \mathcal{H}^{-1}\left(\frac{1-2\beta}{1-\beta}\right)$. However, this is not achieved by an explicit construction of such codes. Here, we presented an explicit construction for $e = 1$ in which the maximum value of t is roughly $\frac{L_M}{\log(L_M)}\frac{1-2\beta}{1-\beta}$. That is, at most only a factor of $\log(L_M)$ from the theoretical upper bound on t.

References

[1] M. Blawat, K. Gaedke, I. Hütter, X.-M. Chen, B. Turczyk, S. Inverso, B.W. Pruitt, and G.M. Church, "Forward error correction for DNA data storage," *Int. Conf. on Computational Science*, vol. 80, pp. 1011–1022, 2016.

[2] J. Bornholt, R. Lopez, D. M. Carmean, L. Ceze, G. Seelig, and K. Strauss, "A DNA-based archival storage system," *Proc. of the Twenty-First Int. Conf. on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, pp. 637–649, Atlanta, GA, Apr. 2016.

[3] Y.M. Chee, H.M. Kiah, and H. Wei, "Efficient and explicit balanced primer codes," https://arxiv.org/abs/1901.01023, 2019.

[4] G. M. Church, Y. Gao, and S. Kosuri, "Next-generation digital information storage in DNA," *Science*, vol. 337, no. 6102, pp. 1628–1628, Sep. 2012.

[5] Y. Erlich and D. Zielinski, "DNA fountain enables a robust and efficient storage architecture," *Science*, vol. 355, no. 6328, pp. 950–954, 2017.

[6] R. Feynman, "There's plenty of room at the bottom," *Engineering and Science, California Institute of Technology*, vol. 23, pp. 22–36, 1960.

[7] D. G. Gibson *et al.*, "Creation of a bacterial cell controlled by a chemically synthesized genome," *Science*, vol. 329, no. 5987, pp. 52–56, Jul. 2010.

[8] N. Goldman, P. Bertone, S. Chen, C. Dessimoz, E. M. LeProust, B. Sipos, and E. Birney, "Towards practical, high-capacity, low-maintenance information storage in synthesized DNA," *Nature*, vol. 494, no. 7435, pp. 77–80, 2013.

[9] R. Heckel, G. Mikutis, and R.N. Grass, "A characterization of the DNA data storage channel," arxiv.org/pdf/1803.03322.pdf, 2018.

[10] S. Jain, F. Farnoud, M. Schwartz, and J. Bruck, "Duplication-correcting codes for data storage in the DNA of living organisms," *IEEE Trans. on Inf. Theory*, vol. 63, no. 8, pp. 4996–5010, Mar. 2017.

[11] A. Lenz, P. H. Siegel, A. Wachter-Zeh, and E. Yaakobi, "Coding over sets for DNA storage," submitted to *IEEE Trans. on Inf. Theory*, Available online https://arxiv.org/abs/1812.02936, 2018.

[12] M. Levy and E. Yaakobi, "Mutually uncorrelated codes for DNA storage," to appear *IEEE Trans. Inform. Theory*.

[13] L. Organick et al. "Scaling up DNA data storage and random access retrieval," *bioRxiv*, Mar. 2017.

[14] C. Rashtchian, K. Makarychev, M. Racz, S. Ang, D. Jevdjic, S. Yekhanin, L. Ceze, and K. Strauss, "Clustering billions of reads for DNA data storage," *NIPS*, 2017.

[15] R.M. Roth, *Introduction to Coding Theory*, Cambridge University Press, 2005.

[16] S. H. T. Yazdi, H. M. Kiah, E. Garcia-Ruiz, J. Ma, H. Zhao, and O. Milenkovic, "DNA-based storage: Trends and methods," *IEEE Transactions on Molecular, Biological and Multi-Scale Communications*, vol 1, no. 3, pp. 230–248, 2015.

[17] S. M. H. T. Yazdi, H. M. Kiah, and O. Milenkovic, "Weakly mutually uncorrelated codes," *IEEE Int. Symp. Inf. Theory*, pp. 2649–2653, Barcelona, Spain, Jul. 2016.

[18] S. M. H. T. Yazdi, Y. Yuan, J. Ma, H. Zhao, and O. Milenkovic, "A rewritable, random-access DNA-based storage system," *Nature Scientific Reports*, vol. 5, no. 14138, Aug. 2015.