Clustering-Correcting Codes

Tal Shinkar, *Student Member, IEEE*, Eitan Yaakobi, *Senior Member, IEEE*, Andreas Lenz, *Member, IEEE*, and Antonia Wachter-Zeh, *Senior Member, IEEE*

***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 the 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 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, their data fields have large distance. 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 symbol of redundancy. The results are first presented for the Hamming metric and are then extended for the edit distance.**

***Index Terms*— DNA, indexes, noise measurement, encoding, decoding, error correction codes, DNA-based storage systems, unordered sequences, clusters.**

I. Introduction

THE idea of using DNA molecules as a volume for storing data was first introduced in 1959 [9]. DNA molecules have the unique qualities of density and durability that make them an attractive solution for storing archivable data. In 1990, the human genome project was initiated with the objective of determining the DNA sequence of the entire human genome, leading to a valuable progress in DNA sequencing and assembly methods. The two main DNA manipulation processes for data storage are synthesis and sequencing. DNA synthesis hereby is the process of creating DNA molecules. Current synthesis methods allow to chemically synthesize arbitrary single-stranded DNA sequences of length a few hundreds [10]. This process results in a DNA pool which consists of all the synthesized strands. Synthesis thus is the process to write data to a DNA storage system. On the other hand, sequencing is the process of reading sequences from the DNA pool.

A rapid growth in synthesis and sequencing technologies has paved the way for the development of non-volatile data storage based on DNA molecules, rendering it a competitive candidate for future archiving technologies. The first large-scale experiments that demonstrated the potential of in vitro DNA storage were reported by Church et al. who recovered 643KB of data [5] and Goldman et al. who accomplished the same task for a 739KB message [11]. However, in both of these works the data was not completely 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], [8], [18], [28]. Other works developed coding solutions which are specifically targeted to correct the special types of errors inside DNA-based storage systems, e.g., [4], [13]–[17], [19], [23], [24], [26]–[28].

In the experiment by Church et al. 10-bit errors occured and Goldman et al. lost two strands of 25 nucleotides. Later, in [29], Grass et al. reported the first system with a usage of error-correcting codes in DNA-based storage and managed to perfectly recover an 81KB message. Bornholt et al. similarly retrieved a 42KB message [2]. Since then, several groups have built similar systems, storing even larger amounts of data. Among these, Erlich and Zielinski [8] stored 2.11MB of data with high storage rate, Blawat et al. [1] successfully stored 22MB, and more recently Organick et al. [18] stored 200MB. Yazdi et al. [28] developed a method that offers both random access and rewritable storage.

A DNA storage system consists of three steps. (see Fig. 1). First, a DNA synthesizer produces strands that contain the encoded data to be stored in DNA. In order to produce strands with an acceptable error rate, the length of the strands is typically limited to no more than 250 nucleotides. The second part is a storage container that stores the DNA strands in an unordered manner. The third part is a DNA sequences that reads back the strands and restores the original, digital, data from them. 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 in an unordered manner, it is unclear what part of the data each strand represents, even if no error occurred. For more details we refer the reader to [12], [16] and referencers therein. Storing DNA strands in a way that will allow to

Manuscript received September 11, 2020; revised August 8, 2021; accepted October 14, 2021. Date of publication November 17, 2021; date of current version February 17, 2022. This work was supported in part by the European Research Council (ERC) through the European Union's Horizon 2020 Research and Innovation Programme under Grant 801434 and in part by the United States–Israel Binational Science Foundation (BSF) under Grant 2018048. An earlier version of this paper was presented in part at the 2019 International Symposium on Information Theory [22] [DOI: 10.1109/ISIT.2019.8849737]. *(Corresponding author: Tal Shinkar.)*

Tal Shinkar and Eitan Yaakobi are with the Computer Science Department, Technion—Israel Institute of Technology, Haifa 3200003, Israel (e-mail: tal.s@cs.technion.ac.il; yaakobi@cs.technion.ac.il).

Andreas Lenz and Antonia Wachter-Zeh are with the Institute for Communications Engineering, Technische Universität München, 80333 Munich, Germany (e-mail: andreas.lenz@mytum.de; antonia.wachter-zeh@tum.de).

Communicated by L. Dolecek, Associate Editor for Coding Techniques.

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TIT.2021.3127174.

Digital Object Identifier 10.1109/TIT.2021.3127174

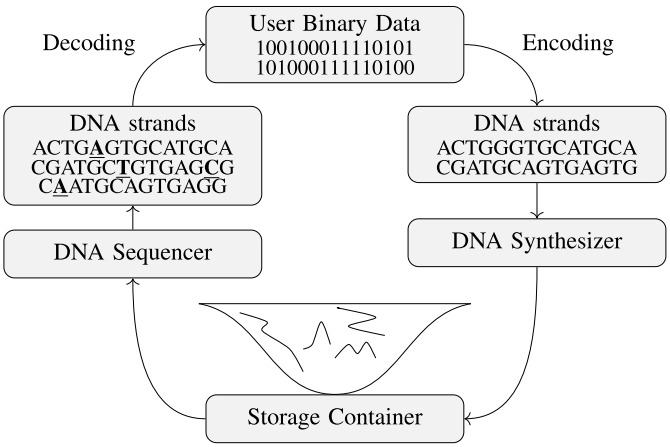

Fig. 1. Illustration of a DNA-based storage system.

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. 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 on the sequencer's input is to partition all the reads into clusters such that all output strands at each cluster are copies of the same information strand. Indices can be handy in this step as well, as a possible solution is to use the indices in order to identify the strands and cluster them together.

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. Also in the presence of errors, clustering based on the index 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. One solution is to add redundancy to the index part in order to correct potential errors in the index. While this technique can be used for random-access [3], this will incur an unavoidable reduction in the storage rate of the DNA storage system.

In this paper a new coding scheme, called *clustering-correcting codes*, is presented which enables to cluster all strands in the right clusters even with the presence of errors in the indices (see Fig. 2), while the redundancy is minimized. We will show how clustering-correcting codes can also be used to correct the errors in the indices. In fact, for a large range of parameters clustering and index-correction can be done with only a single symbol of redundancy for all the strands together.

The rest of the paper is organized as follows. In Section II, some useful definitions that will be used throughout the paper are presented. In Section III, the family of clustering-correcting codes and their capabilities are presented. In Section IV, we present explicit and asymptotic lower and upper bounds on the size of clustering-correcting codes. In Section V, we present an explicit construction of these codes which uses only a single symbol of redundancy. In Section VI, we describe the modifications needed to extend our solution to work under the edit distance as well. Lastly, Section VII concludes the paper.

II. Definitions and Preliminaries

The following notation will be used throughout the paper. For a positive integer n, the set $\{0, 1, \ldots, n-1\}$ is denoted by $[n]$. Let $[q]$ be the alphabet $\{0, 1, \ldots, q-1\}$. For two vectors $\boldsymbol{x}, \boldsymbol{y} \in [q]^n$ 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 $\boldsymbol{x}_{[i,\ell]}$. We also denote the length of the vector $\boldsymbol{x}$ by $|\boldsymbol{x}|$. The Hamming distance between $\boldsymbol{x}$ and $\boldsymbol{y}$ is denoted by $\mathrm{d_H}(\boldsymbol{x}, \boldsymbol{y})$ and the weight of the symbol $s \in [q]$ in $\boldsymbol{x}$ is $w_{\#s}(\boldsymbol{x}) \triangleq |\{i | x_i = s\}|$. The radius-$r$ ball of a vector $\boldsymbol{x} \in [q]^n$ is $B_r(\boldsymbol{x}) = \{\boldsymbol{y} \mid \mathrm{d_H}(\boldsymbol{x}, \boldsymbol{y}) \leqslant r\}$. Since the size of the ball $B_r(\boldsymbol{x})$ does not depend on the choice of $\boldsymbol{x}$ in the Hamming metric we denote this size by $B_r(n) \triangleq \sum_{i=0}^{r} \binom{n}{i}(q-1)^i$ where n denotes the length of $\boldsymbol{x}$. The function

$$\mathcal{H}_q(x) = x\log_q(q-1) - x\log_q(x) - (1-x)\log_q(1-x)$$

for $0 \leqslant x \leqslant 1$ denotes the q-ary entropy function, and the inverse function $\mathcal{H}_q^{-1}(x)$ for $0 \leqslant x \leqslant 1$ is defined to return values between 0 and $\frac{q-1}{q}$.

Assume M strands are stored in a DNA-based storage system where the length of every strand is L. We will assume that $M = q^{\beta L}$ for some $0 < \beta < 1$ and for simplicity, it is assumed that βL is integer. For any integer $i \in [M]$, its q-ary representation of length $\log_q(M)$ is denoted by ind_i. Every length-L strand $\boldsymbol{s}$ that will be stored in the system is of the form $\boldsymbol{s} = (\mathsf{ind}, \boldsymbol{u})$, where ind is the length-$\log_q(M)$ *index field* of the strand (the representation of a number between 0 and $M-1$ using an alphabet of size q) and $\boldsymbol{u}$ is the *data field* of $L - \log_q(M)$ symbols that are used to store the information or the redundancy of an error-correcting code and other coding schemes. 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 denoted by

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

Codes that can correct errors in such sets have been proposed in [30]. Clearly, as the index fields are unique and the data fields are not required to be distinct, $|\mathcal{X}_{M,L}| = q^{M(L-\log_q(M))}$. 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 set of the form $\mathcal{S} = \{(\mathsf{ind}_0, \boldsymbol{u}_0), \ldots, (\mathsf{ind}_{M-1}, \boldsymbol{u}_{M-1})\}$. Note that, due to indexing, the strands in $\mathcal{S}$ are unique and therefore $\mathcal{S}$ is indeed a set. For shorthand, in the rest of the paper the term $L - \log_q(M) = L(1-\beta)$ 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 $\mathcal{G}$ of some $\mathcal{R}$ strands, called the *output strands*, where $\mathcal{R}$ is significantly larger than M. Each output strand in the set $\mathcal{G}$ is a copy of one of the input strands in $\mathcal{S}$, however with some potential errors.

We consider here only substitution errors while extensions for deletions, insertions, and more generally the edit distance will be analyzed in Section VI of the paper. Formally, we define the channel model as follows.

Definition 1: A DNA-based storage system is called a $(t_{\mathsf{i}}, t_{\mathsf{d}})$-**DNA system** if it satisfies the following property: If the output strand $(\mathsf{ind}', \boldsymbol{u}') \in \mathcal{G}$ is a noisy copy of the input strand $(\mathsf{ind}, \boldsymbol{u}) \in \mathcal{S}$, then $\mathrm{d}_{\mathsf{H}}(\mathsf{ind}, \mathsf{ind}') \leqslant t_{\mathsf{i}}$ and $\mathrm{d}_{\mathsf{H}}(\boldsymbol{u}, \boldsymbol{u}') \leqslant t_{\mathsf{d}}$.

That is, the index field has at most t_{i} substitutions while the data field has at most t_{d} substitutions. Since the set $\mathcal{G}$ 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 $\mathcal{R}$ 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 [12]. Another approach to cluster the strands is based upon the distances between pairs of output strands, as was studied in [19]. This approach sorts the strands into buckets based on hash values and then iteratively merges clusters based on the similarity of some representatives.

III. Clustering-Correcting Codes

In this work, we propose a different approach that allows accurate clustering based on indices. In contrast to common clustering methods, we use the fact that it is possible to influence the original strands by encoding them in a carefully chosen manner. We will design the sequences such that we can perform time-efficient clustering described in the following. First, we cluster the output strands based on the indices of the output strands. Then, we scan for output strands which were mis-clustered, that is, were placed in the wrong cluster because of errors in their index field. This is accomplished by checking the distances between the output strands in every cluster in order to either remove completely 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 small complexity. In fact, there is no need to take into account all pairs of strands in a cluster for this operation to succeed. That is, the complexity of this step can be reduced even more. 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. Note that our approach is different from previous work in several aspects. First, while most clustering methods rely on (pseudo-)randomness of the stored DNA strands for good clustering properties, we encode the original strands to ensure that the strands will have properties that allow accurate and fast clustering. Further, to the best of our knowledge, this is the first work that provides guarantees on perfect clustering based on an adversarial channel model.

We proceed by formally defining clustering accuracy. Let $C_{i_0}, \ldots, C_{i_{M-1}}$ be an arbitrary partition of the output strands $\mathcal{G}$ into M clusters. Also let $C'_0, \ldots, C'_{M-1}$, $k \in [M]$ be the correct clusters, i.e., C'_k is the cluster of all strands that are noisy copies of $(\mathsf{ind}_k, \boldsymbol{u}_k)$. We define two types of clustering accuracy: *accurate-clustering* and *complete-clustering*.

Definition 2 (Accurate Clustering): A partition is an **accurate-clustering** if the multisets of clusters agree, i.e., there exists a perfect matching between the partitioned $\{C_{i_0}, \ldots, C_{i_{M-1}}\}$ clusters and the correct clusters $\{C'_0, \ldots, C'_{M-1}\}$.

Remark 1: Upon reading the strands, we read multiple copies of each strand, that may be noisy. Hence, it is possible that the exact same strand appears more than once in a cluster. For this reason, a cluster C_i is represented by a multiset.

Definition 3 (Complete Clustering): A partition is a **complete clustering** if it is an accurate clustering and also $i_h = h$ for all $h \in [M]$. That is, the original index of each output strand is known.

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. 2. 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 constrained codes.

Definition 4 ($(\tau_{\mathsf{i}}, \tau_{\mathsf{d}})$-Clustering Constraint): A set $\mathcal{S} = \{(\mathsf{ind}_0, \boldsymbol{u}_0), \ldots, (\mathsf{ind}_{M-1}, \boldsymbol{u}_{M-1})\} \in \mathcal{X}_{M,L}$ is said to satisfy the $(\tau_{\mathsf{i}}, \tau_{\mathsf{d}})$-**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 $\mathrm{d}_{\mathsf{H}}(\mathsf{ind}_i, \mathsf{ind}_j) \leqslant \tau_{\mathsf{i}}$, it holds that $\mathrm{d}_{\mathsf{H}}(\boldsymbol{u}_i, \boldsymbol{u}_j) \geqslant \tau_{\mathsf{d}}$.

A code $\mathcal{C} \subseteq \mathcal{X}_{M,L}$ is called an $(\tau_{\mathsf{i}}, \tau_{\mathsf{d}})$-**clustering-correcting code** (**CCC**) if every $\mathcal{S} \in \mathcal{C}$ satisfies the $(\tau_{\mathsf{i}}, \tau_{\mathsf{d}})$-clustering constraint.

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

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

Remark 2: A similar name, *Cluster-Correcting Codes* has been used in other works for codes which are capable to correct a cluster of errors in 2-dimensional arrays [21]. In this work we are dealing with a different kind of clustering, regarding partitioning DNA strands into groups defined by their original data.

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

Before we state our main results about the clustering properties of CCCs, we introduce two properties on the clusters after the index-based clustering. Under these conditions, we achieve even stronger clustering properties. The first property is called *majority property* and is defined as follows. Let the cluster C_i

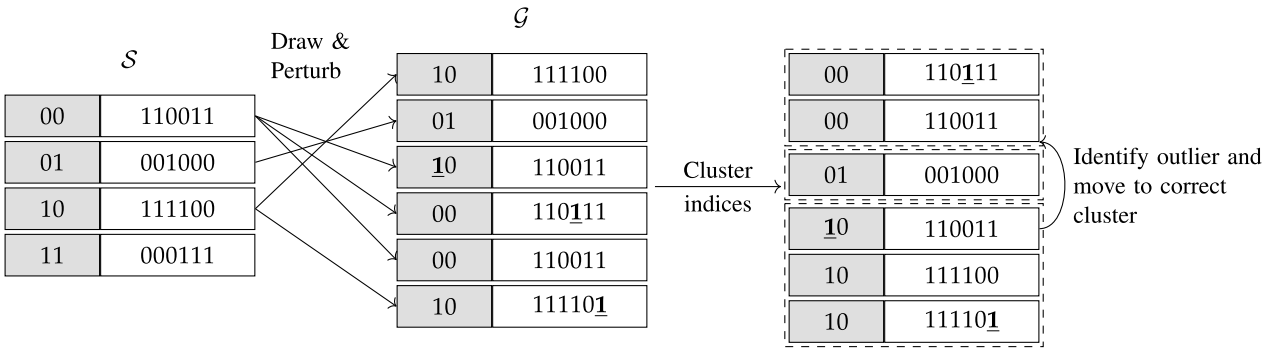

Fig. 2. Exemplary realization of the DNA channel model. A set $\mathcal{S}$ of $M = 4$ binary strands is stored and $\mathcal{R} = 6$ strands are drawn with errors (highlighted in bold). The strands are clustered according to their indices. The outlier can be identified as it has large distance w.r.t. all other strands in the cluster and be put into the correct cluster.

be the group of all strands read with index ind_i (with potential errors in the index).

Definition 5 (Majority Property): Let $C = \{C'_0, \ldots, C'_{M-1}\}$ be a clustering of strands and let $j \in [M]$. Denote by X_j the number of strands in C_j whose index has been received without errors.We say that the set C satisfies the **majority property** if it holds that $X_j \geqslant \frac{|C_j|}{2}$ for all $j \in [M]$. In other words, A clustering set has the **majority property** if in every cluster the majority of the strands have the correct index, i.e., they have no error in the index.

Alternatively, we will discuss a weaker property, which will be referred to as the *dominance property*. Assume we assign a color to each strand in C_i based on the index this strand originated from, so strands that originated from the same index receive the same color.

Definition 6 (Dominance Property): Let $C = \{C'_0, \ldots, C'_{M-1}\}$ be a clustering of strands and let $j \in [M]$. For all $i \in [|C_j|]$, an element $s_i \in C_j$ is of the form $(\mathsf{ind}_j, u'_{k_i})$ where $k_i \in [M]$. Every such strand $s_i = (\mathsf{ind}_j, u'_{k_i})$ is a noisy copy of the strand $(\mathsf{ind}_{k_i}, u_{k_i})$. Denote by X_j the number of strands in C_j whose index has been received without errors. We say that the set C satisfies the **dominance property** if for all $i, j \in [M]$ it holds that $X_j \geqslant |\{(\mathsf{ind}_j, \boldsymbol{u}'_k) \in C_j \mid k = i\}|$. In other words, A clustering set satisfies the **dominance property** if the dominant color, which has the most occurrences, in every cluster C_i is assigned to the strand whose index is correct, which is ind_i.

That is, if a cluster could be partitioned into subsets based on the correct origin of each strand (their true index), the largest subset in the cluster contains the strands with the correct index field which, are therefore clustered correctly. However, partitioning a cluster this way is not necessarily possible in general but we will show how this can be accomplished by using clustering-correcting codes. Lastly, note that majority also implies dominance.

Remark 3: The dominance property can be motivated using the example of a binary symmetric channel (BSC). Assume that the strands are received as a result of transmitting the original strands over a BSC with error probability α. Then, the index of a strand will be received correctly (and thus the strand is clustered correctly) with probability $(1-\alpha)^{\log_q(M)}$. However, a strand from another index will be misclustered into this cluster with probability $(1-\alpha)^{\log_q(M)-d}\alpha^d$, where d is the Hamming distance between the two indices. This probability is smaller by a factor $\left(\frac{\alpha}{1-\alpha}\right)^d$ and therefore, the dominant fraction of strands will be from the correct index.

An expression for the probability that a read cluster satisfies the dominance property is derived as follows. Let $R_0, \ldots, R_{M-1}$ be the number of copies read for each strand encoded in the DNA system. We consider a binary symmetric channel, therefore, $\tau_{\mathsf{i}} = \log_q(M)$, as all the symbols in an index field can be erroneous. We compute the probability that the cluster of the j-th strand satisfy the dominance property. Denote the set of indices of the neighbor clusters of the j-th index with $N_j = \{k | \, \mathrm{d_H}(\mathsf{ind}_j, \mathsf{ind}_k) \leqslant \tau_{\mathsf{i}}, \mathsf{ind}_k \neq \mathsf{ind}_j\}$. $\tau_{\mathsf{i}} = \log_q(M)$ and therefore it holds that $N_j = \{k | 0 \leqslant k < M, k \neq j\}$. Let $k \in N_j$. The probability that a strand from the k-th cluster was misclustered into the j-th cluster is

$$P_k = (1-\alpha)^{\log_q(M) - \mathrm{d_H}(\mathsf{ind}_k, \mathsf{ind}_j)} \alpha^{\mathrm{d_H}(\mathsf{ind}_k, \mathsf{ind}_j)}.$$

Assume for all $k \in N_j$ that X_k represents the number of copies of the k-th strand ended up in the j-th cluster. Also let $X = (X_0, \ldots, X_{j-1}, X_{j+1}, \ldots, X_{M-1})$ The probability that the j-th cluster satisfies the dominance property is

$$D_j(X) = \sum_{i=\mathcal{I}}^{R_j} \binom{R_j}{i} (1-\alpha)^{\log_q(M)\cdot i} \cdot \left(1-(1-\alpha)^{\log_q(M)}\right)^{R_j - i}$$

for $\mathcal{I} = \max_{k \in N_j}\{X_k\} + 1$. Let $\mathcal{X} = [R_0] \times \cdots \times R_{j-1} \times R_{j+1} \times \cdots \times [R_{M-1}]$, and combining these together, the requested probability is given by:

$$\sum_{X \in \mathcal{X}} D_j(X) \cdot \prod_{k \in N_j} \binom{R_k}{X_k} {P_k}^{X_k} \cdot (1-P_k)^{R_k - X_k}.$$

In Figure 3 we plot the probability that a cluster satisfies the dominance property as a function of α.

For the rest of this section, the integers $M, L, q, \tau_{\mathsf{i}}, \tau_{\mathsf{d}}$ are fixed, and $\mathcal{C}$ is an $(\tau_{\mathsf{i}}, \tau_{\mathsf{d}})$-CCC. The next theorem states the clustering capabilities of CCCs.

Theorem 7: Assume that a set $\mathcal{S} \in \mathcal{C}$ is stored in a $(t_{\mathsf{i}}, t_{\mathsf{d}})$-DNA system where $4t_{\mathsf{d}} < \tau_{\mathsf{d}}$. Then, the following properties hold:

1) If $4t_{\mathsf{i}} \leqslant \tau_{\mathsf{i}}$ then accurate clustering of all output strands can be accomplished.

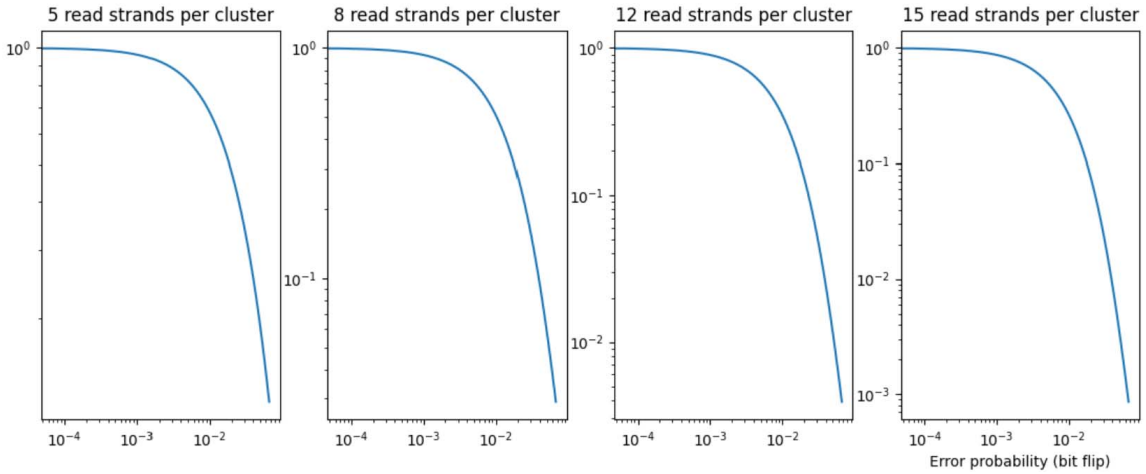

Fig. 3. The probability that a cluster satisfies the dominance property as a function of the probability that a bit is erroneous. The graphs were generated for a DNA-system that contains 1024 strands ($M = 1024$) with a varying number of read strands in each cluster.

2) Under the dominance property, if $2t_{\mathsf{i}} \leqslant \tau_{\mathsf{i}}$ then complete clustering of all output strands can be accomplished.
3) Under the majority property, if $t_{\mathsf{i}} \leqslant \tau_{\mathsf{i}}$ then every mis-clustered output strand, according to its index, can be detected.

Before we prove the theorem, we start with two auxiliary lemmas that will be used in the proof of the theorem.

Lemma 8: Assume that a set $\mathcal{S} \in \mathcal{C}$ is stored in a $(t_{\mathsf{i}}, t_{\mathsf{d}})$-DNA system. For $i \neq j$, let $(\mathsf{ind}'_i, \boldsymbol{u}'_i), (\mathsf{ind}_j, \boldsymbol{u}'_j)$ be a noisy copy of the strand $(\mathsf{ind}_i, \boldsymbol{u}_i), (\mathsf{ind}_j, \boldsymbol{u}_j)$ respectively, such that $\mathsf{ind}'_i = \mathsf{ind}_j$. If $t_{\mathsf{i}} \leqslant \tau_{\mathsf{i}}$ and $4t_{\mathsf{d}} < \tau_{\mathsf{d}}$, it holds that $\mathrm{d}_{\mathsf{H}}(\boldsymbol{u}'_i, \boldsymbol{u}'_j) > 2t_{\mathsf{d}}$.

Proof: Since the data is stored in a $(t_{\mathsf{i}}, t_{\mathsf{d}})$-DNA system, it holds that $\mathrm{d}_{\mathsf{H}}(\mathsf{ind}_i, \mathsf{ind}'_i) = \mathrm{d}_{\mathsf{H}}(\mathsf{ind}_i, \mathsf{ind}_j) \leqslant t_{\mathsf{i}} \leqslant \tau_{\mathsf{i}}$ and $\mathrm{d}_{\mathsf{H}}(\boldsymbol{u}_i, \boldsymbol{u}'_i) \leqslant t_{\mathsf{d}}$. From the fact that $\mathcal{S} \in \mathcal{C}$ we derive that $\mathrm{d}_{\mathsf{H}}(\boldsymbol{u}_i, \boldsymbol{u}_j) \geqslant \tau_{\mathsf{d}} > 4t_{\mathsf{d}}$, and thus $\mathrm{d}_{\mathsf{H}}(\boldsymbol{u}_j, \boldsymbol{u}'_i) \geqslant \mathrm{d}_{\mathsf{H}}(\boldsymbol{u}_i, \boldsymbol{u}_j) - \mathrm{d}_{\mathsf{H}}(\boldsymbol{u}_i, \boldsymbol{u}'_i) > 3t_{\mathsf{d}}$. Also, $\mathrm{d}_{\mathsf{H}}(\boldsymbol{u}_j, \boldsymbol{u}'_j) \leqslant t_{\mathsf{d}}$ which yields that $\mathrm{d}_{\mathsf{H}}(\boldsymbol{u}'_i, \boldsymbol{u}'_j) \geqslant \mathrm{d}_{\mathsf{H}}(\boldsymbol{u}_j, \boldsymbol{u}'_i) - \mathrm{d}_{\mathsf{H}}(\boldsymbol{u}_j, \boldsymbol{u}'_j) > 2t_{\mathsf{d}}$. ∎

Lemma 9: Assume that a set $\mathcal{S} \in \mathcal{C}$ is stored in a $(t_{\mathsf{i}}, t_{\mathsf{d}})$-DNA system. Let $(\mathsf{ind}'_j, \boldsymbol{u}'_j), (\mathsf{ind}'_k, \boldsymbol{u}'_k)$ be a noisy copy of the strands $(\mathsf{ind}_j, \boldsymbol{u}_j), (\mathsf{ind}_k, \boldsymbol{u}_k)$ respectively, such that $\mathsf{ind}'_j = \mathsf{ind}'_k$. If $2t_{\mathsf{i}} \leqslant \tau_{\mathsf{i}}$ and $4t_{\mathsf{d}} < \tau_{\mathsf{d}}$, it holds that $\mathrm{d}_{\mathsf{H}}(\boldsymbol{u}'_j, \boldsymbol{u}'_k) \leqslant 2t_{\mathsf{d}}$ if and only if $\mathsf{ind}_j = \mathsf{ind}_k$.

Proof: It holds that $\mathrm{d}_{\mathsf{H}}(\mathsf{ind}_j, \mathsf{ind}'_j) \leqslant t_{\mathsf{i}} \leqslant \frac{\tau_{\mathsf{i}}}{2}$ and also, $\mathrm{d}_{\mathsf{H}}(\mathsf{ind}_k, \mathsf{ind}'_k) \leqslant t_{\mathsf{i}} \leqslant \frac{\tau_{\mathsf{i}}}{2}$. As $\mathsf{ind}'_j = \mathsf{ind}'_k$ it holds that, $\mathrm{d}_{\mathsf{H}}(\mathsf{ind}_j, \mathsf{ind}_k) \leqslant \tau_{\mathsf{i}}$, and therefore they need to satisfy the clustering constraint, that is, for $\mathsf{ind}_j \neq \mathsf{ind}_k$ it holds that $\mathrm{d}_{\mathsf{H}}(\boldsymbol{u}_j, \boldsymbol{u}_k) \geqslant \tau_{\mathsf{d}}$. Similarly to Lemma 8 it holds that $\mathrm{d}_{\mathsf{H}}(\boldsymbol{u}'_j, \boldsymbol{u}'_k) > 2t_{\mathsf{d}}$. Hence, if $\mathrm{d}_{\mathsf{H}}(\boldsymbol{u}'_j, \boldsymbol{u}'_k) \leqslant 2t_{\mathsf{d}}$ then $\mathsf{ind}_j = \mathsf{ind}_k$. For the opposite direction, if $\mathsf{ind}_j = \mathsf{ind}_k$, since the data is stored in a (t_i, t_{d})-DNA system, it holds that $\mathrm{d}_{\mathsf{H}}(\boldsymbol{u}_j, \boldsymbol{u}'_j) \leqslant t_{\mathsf{d}}$ and $\mathrm{d}_{\mathsf{H}}(\boldsymbol{u}_k, \boldsymbol{u}'_k) \leqslant t_{\mathsf{d}}$. Therfore $\mathrm{d}_{\mathsf{H}}(\boldsymbol{u}'_j, \boldsymbol{u}'_k) \leqslant 2t_{\mathsf{d}}$. ∎

Proof of Theorem 7: For convenience we prove the claims at the opposite order.

3) Let $(\mathsf{ind}'_i, \boldsymbol{u}'_i)$ be a noisy copy of the strand $(\mathsf{ind}_i, \boldsymbol{u}_i)$ and $j \neq i \in [M]$ such that $\mathsf{ind}_j = \mathsf{ind}'_i$. Let $(\mathsf{ind}_j, \boldsymbol{u}'_j)$ be a noisy copy of the strand $(\mathsf{ind}_j, \boldsymbol{u}_j)$, that is, errors might occur in the data field but not in the index field. Thus, from Lemma 8, it holds that $\mathrm{d}_{\mathsf{H}}(\boldsymbol{u}'_i, \boldsymbol{u}'_j) > 2t_{\mathsf{d}}$. On the other hand, the distance between the data fields of the two strands that belong to the same cluster is at most $2t_{\mathsf{d}}$ as the data fields reside in the same radius-t_{d} ball of the original data. That is, under the majority property, a mis-clustered strand will have a distance larger than $2t_{\mathsf{d}}$ from the majority of the strands in the cluster, and so it can be removed instead of being mis-clustered. All remaining strands have a correct index field and are thus placed in their correct cluster.

2) Let $(\mathsf{ind}'_i, \boldsymbol{u}'_i), (\mathsf{ind}'_k, \boldsymbol{u}'_k)$ be a noisy copy of $(\mathsf{ind}_i, \boldsymbol{u}_i)$, $(\mathsf{ind}_k, \boldsymbol{u}_k)$ that ended up in the same cluster of the j-th strand, respectively. That is, $\mathsf{ind}'_i = \mathsf{ind}'_k = \mathsf{ind}_j$. According to Lemma 9 it is deduced that $\mathrm{d}_{\mathsf{H}}(\boldsymbol{u}'_i, \boldsymbol{u}'_k) \leqslant 2t_{\mathsf{d}}$ if and only if $\mathsf{ind}_i = \mathsf{ind}_k$. This way, each cluster can be partitioned into mutually disjoint subsets, such that every subset contains copies of the same information strand. At most one of those subsets contains strands with the correct index. Therefore, under the dominance property, it is possible to identify the subset with the correct index, since it is the subset of the largest size. For $i \in [M]$, the subset of index ind_i will be denoted by C_i.

After applying this partitioning to all clusters we seek to move subsets that are not dominant into their correct cluster. Let W be one of those subsets. Also let $s'_i = (\mathsf{ind}'_i, \boldsymbol{u}'_i) \in W$ be a noisy copy of the strand $s_i = (\mathsf{ind}_i, \boldsymbol{u}_i)$. It holds that $\mathrm{d}_{\mathsf{H}}(\mathsf{ind}_i, \mathsf{ind}'_i) \leqslant t_{\mathsf{i}}$, and hence, $\mathsf{ind}_i \in B_{t_{\mathsf{i}}}(\mathsf{ind}'_i)$. In other words $B_{t_{\mathsf{i}}}(\mathsf{ind}'_i)$ is the set of all candidates for the correct index ind_i. As the partitioning to subsets was already completed, for each $\mathsf{ind}_k \in B_{t_{\mathsf{i}}}(\mathsf{ind}'_i)$ there exists some strand $s'_k = (\mathsf{ind}_k, \boldsymbol{u}'_k) \in C_k$. From Lemma 9 it holds that $\mathsf{ind}_k = \mathsf{ind}_i$ if and only if $\mathrm{d}_{\mathsf{H}}(\boldsymbol{u}'_i, \boldsymbol{u}'_k) \leqslant 2t_{\mathsf{d}}$. At this point each cluster contains strands with a single unique index. That is, going through all index candidates and verifying this criteria, exactly one of the candidates will match. Therefore, we can find the correct cluster that the noisy strand s'_i (and the subset it represents) belongs to. Hence, the index fields of all the output strands can be corrected, and the achieved clustering will be complete.

1) We first apply the same partitioning from the proof of 2). This is possible to accomplish since $2t_{\mathsf{i}} < 4t_{\mathsf{i}} \leqslant \tau_{\mathsf{i}}$. Next, we seek to unify subsets that originated with the same index. Let W_i, W_k be a subset of strands read with index field $\mathsf{ind}'_i, \mathsf{ind}'_k$ respectively. Also assume that $\mathsf{ind}'_i, \mathsf{ind}'_k$ is a noisy copy of $\mathsf{ind}_i, \mathsf{ind}_k$ respectively. It holds that $\mathrm{d}_{\mathsf{H}}(\mathsf{ind}'_i, \mathsf{ind}_i) \leqslant t_{\mathsf{i}}$, and $\mathrm{d}_{\mathsf{H}}(\mathsf{ind}'_k, \mathsf{ind}_k) \leqslant t_{\mathsf{i}}$. The subsets W_i, W_k should be unified if and only if $i = k$. Note that in case $i = k$ it holds that $\mathrm{d}_{\mathsf{H}}(\mathsf{ind}'_i, \mathsf{ind}'_k) \leqslant \mathrm{d}_{\mathsf{H}}(\mathsf{ind}_i, \mathsf{ind}'_i) + \mathrm{d}_{\mathsf{H}}(\mathsf{ind}_k, \mathsf{ind}'_k) \leqslant 2t_{\mathsf{i}}$. Therefore, any subset W_k that is a candidate to be unified with a subset W_i satisfies $\mathsf{ind}'_k \in B_{2t_{\mathsf{i}}}(\mathsf{ind}'_i)$. Let W_j be a subset of strands read with index field ind'_j which is a noisy copy of ind_j. Assume W_j is a candidate, hence, $\mathsf{ind}'_j \in B_{2t_{\mathsf{i}}}(\mathsf{ind}'_i)$. That is, $\mathrm{d}_{\mathsf{H}}(\mathsf{ind}'_j, \mathsf{ind}'_i) \leqslant 2t_{\mathsf{i}}$. It holds that

$$\mathrm{d}_{\mathsf{H}}(\mathsf{ind}_i,\mathsf{ind}_j) \leqslant \mathrm{d}_{\mathsf{H}}(\mathsf{ind}_i, \mathsf{ind}'_i) + \mathrm{d}_{\mathsf{H}}(\mathsf{ind}'_i, \mathsf{ind}'_j) + \mathrm{d}_{\mathsf{H}}(\mathsf{ind}'_j, \mathsf{ind}_j),$$

and hence, $\mathrm{d}_{\mathsf{H}}(\mathsf{ind}_i, \mathsf{ind}_j) \leqslant 4t_{\mathsf{i}} \leqslant \tau_{\mathsf{i}}$. In particular, any two strands $s_i = (\mathsf{ind}'_i, \boldsymbol{u}'_i) \in W_i$ and $s_j = (\mathsf{ind}'_j, \boldsymbol{u}'_j) \in W_j$ satisfy the clustering constraint. Furthermore, from Lemma 9, $\mathsf{ind}_i = \mathsf{ind}_j$ if and only if $\mathrm{d}_{\mathsf{H}}(\boldsymbol{u}'_i, \boldsymbol{u}'_j) \leqslant 2t_{\mathsf{d}}$. Therefore, by measuring the distance between their two data fields, it can be deduced whether $\mathsf{ind}_i = \mathsf{ind}_j$. Note that a single comparison is enough to determine if the two subsets should be combined. Repeating the process for every two subsets that might be unified it is possible to produce M clusters $C_{i_0}, \ldots, C_{i_{M-1}}$ such that every cluster C_{i_j} contains exactly the strands that originated from the same input strand. However, the complete mapping between the clusters and the indices is not guaranteed. Hence, it is possible to output an accurate clustering. ■

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. Note that in Theorem 7, while the condition on the relation between t_{i} and τ_{i} is relaxed from the first to the third claim, the requirement on the number of strands with the correct index in a cluster strengthens. Yet, the best clustering output is guaranteed in the second claim of Theorem 7. In this claim we could use the dominance property instead of the stronger majority property as we also assumed that $2t_{\mathsf{i}} \leqslant \tau_{\mathsf{i}}$, which is a necessity to perform the error correction of the indices and thus the clusters. On the other hand, majority implies dominance and therefore this claim works.

The third claim of Theorem 7 guarantees accurate clustering, that is, the index fields of the output strands may still remain erroneous. Yet, it is still possible to reduce the set of candidate indices for each cluster. Namely, let $C_{i_0}, \ldots, C_{i_{M-1}}$ be an accurate clustering. The correct index of the cluster C_{i_j} must belong to the following set of indices

$$\bigcap_{(\mathsf{ind}_k, \boldsymbol{u}_k) \in C_{i_j}} B_{t_{\mathsf{i}}}(\mathsf{ind}_k).$$

The clustering algorithm according to Theorem 7 first partitions the strands to clusters according to the indices and then compares the distances *only* between strands in the same cluster. According to Theorem 7, when reading data stored with the $(\tau_{\mathsf{i}}, \tau_{\mathsf{d}})$-clustering constraint, even if all index fields are erroneous, it is possible to achieve an accurate clustering. In this process strands which are read with the same index are compared to form a partitioning of the reads into subsets. Then, comparisons between representatives of those subsets are performed. The amount of comparisons in this last step is independent of the number of output strands. Hence, the amount of comparisons is minimized significantly with respect to comparing between all pairs of all output strands. In addition, if the dominance property holds, it is also possible to correct all the index fields in all output strands while the complexity is reduced even further.

Algorithm 1 Error Identification for the Majority Property Using the $(\tau_{\mathsf{i}}, \tau_{\mathsf{d}})-$ Clustering Constraint

Input: $\mathcal{R}$ read strands $(\mathsf{ind}_{i_0}, \boldsymbol{u}_0), \ldots, (\mathsf{ind}_{i_{R-1}}, \boldsymbol{u}_{R-1}) \in [q]^L$
Output: Clusters $C_0, \ldots, C_{M-1}$
1: $C_k \leftarrow \{(\mathsf{ind}_{i_j}, \boldsymbol{u}_j) | \mathsf{ind}_{i_j} = \mathsf{ind}_k\}$, $0 \leqslant k < M$
2: **for** $k \leftarrow 0, 1, \ldots, M-1$ **do**
3: $\quad F \leftarrow C_k$
4: $\quad$ **while** $|F| > |C_k|/2$ **do**
5: $\quad\quad$ Pick $(\mathsf{ind}_k, \boldsymbol{u}) \in C_k$
6: $\quad\quad F \leftarrow \{(\mathsf{ind}_k, \boldsymbol{v}) | (\mathsf{ind}_k, \boldsymbol{v}) \in C_k \wedge \mathrm{d}_{\mathsf{H}}(\boldsymbol{v}, \boldsymbol{u}) > 2t_{\mathsf{d}}\}$
7: $\quad\quad$ **if** $|F| \leqslant |C_k|/2$ **then** $C_k \leftarrow C_k \setminus F$ **else** $C_k \leftarrow F$
8: $\quad$ **end while**
9: **end for**

From Theorem 7 we can derive two algorithms. The first one, Algorithm 1, handles identification of errors in the index field of the read strands under the majority property. Algorithm 2, performs also the correction of the erroneous indices and requires only the dominance property.

Assume that a set $\mathcal{S} \in \mathcal{C}$ is stored in a $(t_{\mathsf{i}}, t_{\mathsf{d}})$-DNA system. In addition, let $\mathcal{R}$ be the number of reads upon sequencing.

Theorem 10: Under the majority property, if $t_{\mathsf{i}} \leqslant \tau_{\mathsf{i}}$ and $4t_{\mathsf{d}} < \tau_{\mathsf{d}}$ then the output clusters $C_0, \ldots, C_{M-1}$ of Algorithm 1 contain only strands with the correct index. Furthermore, the complexity of Algorithm 1 is $\mathcal{O}(B_{t_{\mathsf{i}}}(\log_q(M)) \cdot \mathcal{R} \cdot L)$ symbol operations.

Proof: Algorithm 1 starts by iterating over the range of possible indices. For each index k the algorithm first gathers all output strands read with index field ind_k in Step 1. Then the loop in Step 4 removes strands with errors in the index field out of the cluster C_k. Thus, if Algorithm 1 terminates, the clusters $C_0, \ldots, C_{M-1}$ contain only strands with the correct index. In every iteration of the while loop in Step 4 the algorithm picks a strand $(\mathsf{ind}_k, \boldsymbol{u})$ in Step 5 from the cluster and compares it with all other strands in this cluster. In Step 6 the algorithm computes the set of all strands which the Hamming distance between their data field and $\boldsymbol{u}$ is greater than $2t_{\mathsf{d}}$. Lastly, according to Theorem 7, if the picked strand has the correct index then the majority of the strands in the cluster will not belong to the set F, hence $|F| \leqslant |C_k|/2$. Therefore, in Step 7 strands that are not copies of the k-th input strand are detected as erroneous and are removed from the cluster C_k. Note also that when a strand with the correct index is picked, all strands with errors in their index fields must be in F. When picking such a strand, the condition of the while loop in Step 4 no longer holds and the while loop terminates. Furthermore,

Algorithm 2 Error Correction for the Dominance Property Using the $(\tau_{\mathsf{i}}, \tau_{\mathsf{d}})-$ Clustering Constraint

Input: $\mathcal{R}$ read strands $(\mathsf{ind}_{i_0}, \boldsymbol{u}_0), \ldots, (\mathsf{ind}_{i_{\mathcal{R}-1}}, \boldsymbol{u}_{\mathcal{R}-1}) \in [q]^L$

Output: Clusters $C_0, \ldots, C_{M-1}$

1: $\mathcal{I} \leftarrow \emptyset$
2: $C_k \leftarrow \{(\mathsf{ind}_j, \boldsymbol{u}) | \mathsf{ind}_j = \mathsf{ind}_k\}$, $0 \leqslant k < M$
3: **for** $i = 0, 1, \ldots, M-1$ **do**
4: $\quad P \leftarrow \{\{(\mathsf{ind}_j, \boldsymbol{v}) | \, \mathrm{d}_{\mathsf{H}}(\boldsymbol{v}, \boldsymbol{u}) \leqslant 2t_{\mathsf{d}}\} | (\mathsf{ind}_i, \boldsymbol{u}) \in C_k\}$
5: $\quad C_k \leftarrow \underset{C \in P}{\operatorname{argmax}} |C|$
6: $\quad \mathcal{I} \leftarrow \mathcal{I} \cup (P \setminus C_k)$
7: **end for**
8: **for** $W \in \mathcal{I}$, let $(\mathsf{ind}_i, \boldsymbol{u}) \in W$ **do**
9: $\quad$ **for** $\mathsf{ind}_j \in B_{t_{\mathsf{i}}}(\mathsf{ind}_i)$, let $(\mathsf{ind}_j, \boldsymbol{v}'_j) \in C_j$ **do**
10: $\quad\quad$ **if** $\mathrm{d}_{\mathsf{H}}(\boldsymbol{v}'_j, \boldsymbol{u}) \leqslant 2t_{\mathsf{d}}$ **then**
11: $\quad\quad\quad C_j \leftarrow C_j \cup \{(\mathsf{ind}_j, \boldsymbol{v}) | (\mathsf{ind}_i, \boldsymbol{v}) \in W\}$
12: $\quad\quad$ **end if**
13: $\quad$ **end for**
14: **end for**

a strand with the correct index will be picked after at most $|C_k|/2$ iterations and therefore the algorithm terminates.

Next, for the complexity notice that Step 1 can be done with one pass over all strands by reading the indices and moving the strands to the matching clusters. Let $\mathcal{I}' = \{\mathsf{ind}_{i_1}, \ldots, \mathsf{ind}_{i_k}\}$ be the set of the correct indices of the different strands in C_k. In each iteration of the while loop in Step 4 we remove from C_k at least one of the indices in $\mathcal{I}'$. This is true since all strands in C_k which originated from the same input strand will be removed together in Step 5. Since $|\mathcal{I}'| \leqslant B_{t_{\mathsf{i}}}(\log_q(M))$ we can bound the number of iterations needed to successfully complete the identification of the erroneous indices with $B_{t_{\mathsf{i}}}(\log_q(M))$. At the end of this process each cluster contains only the strands that were read with an error-free index field. Denote by $C_0, C_1, \ldots, C_{M-1}$ the different read clusters after their initialization in Step 1, so it holds that $\sum_{i=0}^{m-1} C_k = \mathcal{R}$. The comparisons performed in Step 6 requires $|C_k| \cdot L$ symbol operations. Therefore, the number of symbol operations required to identify errors in the k-th cluster is

$$O(B_{t_{\mathsf{i}}}(\log_q(M)) \cdot |C_k| \cdot L).$$

Summing up for all clusters, the overall complexity of the algorithm is

$$O(B_{t_{\mathsf{i}}}(\log_q(M)) \cdot \mathcal{R} \cdot L).$$

■

Next, we present the second algorithm derived from Theorem 7, which can correct errors in the index fields and provide a complete clustering.

Theorem 11: Under the dominance property, if $t_{\mathsf{i}} \leqslant \tau_{\mathsf{i}}/2$ and $4t_{\mathsf{d}} < \tau_{\mathsf{i}}$ then Algorithm 2 outputs clusters $C_0, \ldots, C_{M-1}$ such that all strands in all clusters have correct index fields. Furthermore, its complexity is

$$O(B_{t_{\mathsf{i}}}(\log_q(M)) \cdot L \cdot (\mathcal{R} + B_{t_{\mathsf{i}}}(\log_q(M)) \cdot M))$$

symbol operations.

Proof: Algorithm 2 starts by detecting which strands are mis-clustered. The error identification process ends in Step 6. Theorem 7 states that two strands originated from the same index if and only if the Hamming distance of the data fields between the two strands is at most $2t_{\mathsf{d}}$. Therefore, in Step 4 the cluster is partitioned into subsets based on the distances between the data fields of all strands. Then the largest set in the partition is selected as C_k in Step 5. In Step 6 the other sets in the partition are stored in $\mathcal{I}$ so we can correct their indices later on.

The partition of C_k can be done with $O(B_{t_{\mathsf{i}}}(\log_q(M)) \cdot |C_k|)$ comparisons. This is done by picking a strand $(\mathsf{ind}_k, \boldsymbol{u}) \in C_k$ and computing the set F as done in Step 6 of Algorithm 1. The set $C_k \setminus F$ is exactly the subset in the partition that contains $(\mathsf{ind}_k, \boldsymbol{u})$. Next, partition F recursively until there are $B_{t_{\mathsf{i}}}(\log_q(M))$ subsets in the partition. Hence, $B_{t_{\mathsf{i}}}(\log_q(M))$ iterations. This process requires

$$O(B_{t_{\mathsf{i}}}(\log_q(M)) \cdot |C_k| \cdot L)$$

symbol operations per cluster, and a total of

$$O(B_{t_{\mathsf{i}}}(\log_q(M)) \cdot \mathcal{R} \cdot L)$$

for all clusters together.

In Steps 8 to 11 the algorithm performs error correction in the index fields of the erroneous strands. For every subset $W \in \mathcal{I}$, a strand is picked to be the representative of this subset. Next its data field is compared against a strand from each of the clusters that their index is in $B_{t_{\mathsf{i}}}(\mathsf{ind}_i))$. From Theorem 7, the representative must have been originated in one of those clusters. Thus, the single cluster C_j in $B_{t_{\mathsf{i}}}(\mathsf{ind}_i))$ that will match in Step 10 is the correct cluster for W.

The correction process requires $B_{t_{\mathsf{i}}}(\log_q(M))$ comparisons for each set in $\mathcal{I}$. The size of $\mathcal{I}$ is at most $M \cdot B_{t_{\mathsf{i}}}(\log_q(M))$ as the size of the partition in Step 4 is at most $B_{t_{\mathsf{i}}}(\log_q(M))$. That is, the error correction part requires

$$O((B_{t_{\mathsf{i}}}(\log_q(M)))^2 \cdot M \cdot L)$$

symbol operations. Lastly, it is concluded that the overall complexity of Algorithm 2 is

$$O(B_{t_{\mathsf{i}}}(\log_q(M)) \cdot L \cdot (\mathcal{R} + B_{t_{\mathsf{i}}}(\log_q(M)) \cdot M).$$

■

IV. Bounds

In this section, upper and lower bounds on the size of the largest $(\tau_{\mathsf{i}}, \tau_{\mathsf{d}})$-CCC $A_{M,L}(\tau_{\mathsf{i}}, \tau_{\mathsf{d}})$ are presented. Recall that $L_M = L(1-\beta)$ and let $C_n(d)$ be the size of the largest length-n error-correcting code $C \subseteq [q]^n$ with minimum Hamming distance d. For the rest of the section, let $B_1 = B_{\tau_{\mathsf{i}}}(\log_q(M)) - 1, B_2 = B_{\tau_{\mathsf{d}}-1}(L_M)$, $\mathsf{le}_q = \log_q(\exp(1))$, and it is also assumed that $\beta < 1/2$. For the rest of this section we assume that $B_1 \cdot B_2 < q^{L_M}$.

We start with an upper bound on the redundancy of CCCs.

Theorem 12: For all $M, L, q, \tau_{\mathsf{i}}$, and τ_{d} it holds that

$$A_{M,L}(\tau_{\mathsf{i}}, \tau_{\mathsf{d}}) \geqslant q^{ML_M} \left(1 - \frac{B_1 B_2}{q^{L_M}}\right)^{M-E}$$

and hence

$$r_{M,L}(\tau_{\mathsf{i}},\tau_{\mathsf{d}}) < \frac{\mathsf{le}_q \cdot (M-E)B_1B_2}{q^{L_M}-B_1B_2},$$

where $E = C_{\log_q(M)}(\tau_{\mathsf{i}}+1)$.

Proof: In order to verify the lower bound, we construct an $(\tau_{\mathsf{i}},\tau_{\mathsf{d}})$-CCC $\mathcal{C}$ that will yield a lower bound on $A_{M,L}(\tau_{\mathsf{i}},\tau_{\mathsf{d}})$. Let $C_1 \subseteq [q]^{\log_q(M)}$ be a length-$\log_q(M)$ error-correcting code with minimum Hamming distance $\tau_{\mathsf{i}}+1$ and cardinality $E = |C_1| = C_{\log_q(M)}(\tau_{\mathsf{i}}+1)$.

Each codeword $\mathcal{S} = \{(\mathsf{ind}_0,\boldsymbol{u}_0),\ldots,(\mathsf{ind}_{M-1},\boldsymbol{u}_{M-1})\} \in \mathcal{C}$ 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 q^{L_M} options for each strand and thus $(q^{L_M})^E$ options for the first step. Since the Hamming distance between all pairs of indices of these strands is at least $\tau_{\mathsf{i}}+1$, 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-$(\tau_{\mathsf{d}}-1)$ balls are mutually disjoint. Therefore we exclude all the data vectors in the balls of all the neighboring indices. Thus, there are at least

$$q^{L_M} - (B_{\tau_{\mathsf{i}}}(\log_q(M))-1)\cdot B_{\tau_{\mathsf{d}}-1}(L_M) = q^{L_M} - B_1B_2$$

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

$$A_{M,L}(\tau_{\mathsf{i}},\tau_{\mathsf{d}}) \geqslant q^{ML_M}\left(1-\frac{B_1B_2}{q^{L_M}}\right)^{M-E}.$$

We can also deduce an upper bound on the redundancy

$$\begin{aligned} r_{M,L}(\tau_{\mathsf{i}},\tau_{\mathsf{d}}) &= M\cdot L_M - \log_q(A_{M,L}(\tau_{\mathsf{i}},\tau_{\mathsf{d}})) \\ &\leqslant -(M-E)\cdot\log_q\left(1-\frac{B_1B_2}{q^{L_M}}\right) \\ &< \frac{\mathsf{le}_q\cdot(M-E)B_1B_2}{q^{L_M}-B_1B_2}, \end{aligned}$$

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

The next corollary follows directly from Theorem 12.

Corollary 13: For $M = q^{\beta L}$ and all τ_{d} such that

$$\tau_{\mathsf{d}} \leqslant L_M\mathcal{H}_q^{-1}\left(\frac{1-2\beta}{1-\beta} - \frac{\log_q(\mathsf{le}_q)}{(1-\beta)L} + \frac{\beta}{1-\beta}\cdot\mathcal{H}_q\left(\frac{\tau_{\mathsf{i}}}{\beta L}\right)\right)$$

it holds that $r_{M,L}(\tau_{\mathsf{i}},\tau_{\mathsf{d}}) < 1$.

Proof: From Theorem 12 it holds that

$$r_{M,L}(\tau_{\mathsf{i}},\tau_{\mathsf{d}}) < \frac{\mathsf{le}_q\cdot(M-E)B_1B_2}{q^{L_M}-B_1B_2},$$

where $E = C_{\log_q(M)}(\tau_{\mathsf{i}}+1)$. Hence it is enough to prove that $B_1B_2 \leqslant \frac{q^{L_M}}{\mathsf{le}_q\cdot(M-E)+1}$. According to Lemma 4.7 in [20], $B_r(n) \leqslant q^{n\cdot\mathcal{H}_q\left(\frac{r}{n}\right)}$ and so it is sufficient to show that $q^{L_M\mathcal{H}_q\left(\frac{\tau_{\mathsf{d}}-1}{L_M}\right)} \leqslant \frac{q^{L_M}}{B_1(\mathsf{le}_q\cdot(M-E)+1)}$. Now, as $M = q^{\beta L}$ and we assumed

$$\tau_{\mathsf{d}} \leqslant L_M\mathcal{H}_q^{-1}\left(\frac{1-2\beta}{1-\beta} - \frac{\log_q(\mathsf{le}_q)}{(1-\beta)L} + \frac{\beta}{1-\beta}\cdot\mathcal{H}_q\left(\frac{\tau_{\mathsf{i}}}{\beta L}\right)\right),$$

the following holds

$$\begin{aligned} \mathcal{H}_q\left(\frac{\tau_{\mathsf{d}}-1}{L_M}\right) &\leqslant \frac{1-2\beta}{1-\beta} - \frac{\log_q(\mathsf{le}_q)}{(1-\beta)L} + \frac{\beta}{1-\beta}\cdot\mathcal{H}_q\left(\frac{\tau_{\mathsf{i}}}{\beta L}\right) \\ &= \frac{1}{L_M}\cdot\left(L_M - \log_q(M) - \log_q(\mathsf{le}_q) - \log_q(M)\cdot\mathcal{H}_q\left(\frac{\tau_{\mathsf{i}}}{\log_q(M)}\right)\right) \\ &= \frac{1}{L_M}\cdot\left(L_M - \log_q(M\cdot\mathsf{le}_q) - \log_q(M)\cdot\mathcal{H}_q\left(\frac{\tau_{\mathsf{i}}}{\log_q(M)}\right)\right) \\ &\leqslant \frac{1}{L_M}\cdot\left(L_M - \log_q(M\cdot\mathsf{le}_q) - \log_q(B_1)\right) \\ &\leqslant \frac{1}{L_M}\cdot\left(L_M - \log_q(1+\mathsf{le}_q\cdot(M-E)) - \log_q(B_1)\right). \end{aligned}$$

Hence,

$$q^{L_M\mathcal{H}_q\left(\frac{\tau_{\mathsf{d}}-1}{L_M}\right)} \leqslant \frac{q^{L_M}}{B_1(\mathsf{le}_q\cdot(M-E)+1)}.$$

∎

A similar expression for a lower bound on $r_{M,L}(\tau_{\mathsf{i}},\tau_{\mathsf{d}})$ is presented next.

Theorem 14: For all M, L, τ_{i} and τ_{d} it holds that

$$A_{M,L}(\tau_{\mathsf{i}},\tau_{\mathsf{d}}) \leqslant q^{M\cdot L_M}\left(1-\frac{B_2}{q^{L_M}}\right)^{M-1},$$

and hence

$$r_{M,L}(\tau_{\mathsf{i}},\tau_{\mathsf{d}}) > \frac{\mathsf{le}_q\cdot(M-1)\cdot B_2}{q^{L_M}}.$$

Proof: Let $\mathcal{C}$ be an $(\tau_{\mathsf{i}},\tau_{\mathsf{d}})$-CCC of maximal size $A_{M,L}(\tau_{\mathsf{i}},\tau_{\mathsf{d}})$ and let $\mathcal{S} \in \mathcal{C}$ be a codeword. For two strands $(\mathsf{ind}_i,\boldsymbol{u}_i),(\mathsf{ind}_j,\boldsymbol{u}_j)$ we say that they are τ_{i}-*neighbors* if $\mathrm{d}_{\mathsf{H}}(\mathsf{ind}_i,\mathsf{ind}_j) \leqslant \tau_{\mathsf{i}}$. By definition, the codeword $\mathcal{S} \in \mathcal{C}$ satisfies the $(\tau_{\mathsf{i}},\tau_{\mathsf{d}})$-clustering constraint, and therefore each strand $(\mathsf{ind}_i,\boldsymbol{u}_i) \in \mathcal{S}$ satisfies the constraint with respect to its τ_{i}-neighbors. That is, for each τ_{i}-neighbor strand $(\mathsf{ind}_j,\boldsymbol{u}_j) \in \mathcal{S}$ it holds that $\mathrm{d}_{\mathsf{H}}(\boldsymbol{u}_i,\boldsymbol{u}_j) \geqslant \tau_{\mathsf{d}}$, or equivalently $\boldsymbol{u}_i \notin B_{\tau_{\mathsf{d}}}(\boldsymbol{u}_j)$. Next, we list all possible codewords in $\mathcal{C}$, and deduce an upper bound on the cardinality of $\mathcal{C}$. This is done by going through the indices of the M strands in each codeword, ordered by their weight, starting from zero. The order in every group of indices with the same weight is arbitrary. For the first strand, any length-L_M vector can be assigned for its data field, without violating the clustering constraint, and hence, there are at most q^{L_M} options. Then, for each strand s_i there are $q^{L_M} - k_i\cdot B_{\tau_{\mathsf{d}}}(L_M)$ options to assign for its data field, where k_i is defined as $k_i = |\{\boldsymbol{u}_j | s_j = (\mathsf{ind}_j,\boldsymbol{u}_j)$ and s_j is an already assigned τ_{i}-neighbor of $s_i\}|$. It is safe to say that $k_i \geqslant 1$ for all i without miscounting any valid codeword. This is true as $\tau_{\mathsf{i}} \geqslant 1$ (for $\tau_{\mathsf{i}} = 0$ we get trivially $\mathcal{C} = \mathcal{X}_{M,L}$) and there always exists a 1-neighbor that already has its data-field assigned. The case of $k_i = 1$ refers to the scenario where for each s_i all its τ_{i}-neighbors were assigned with identical data. For this reason the resulting bound does not dependent on τ_{i}.

Hence, for each strand, besides the first one, we have at most $q^{L_M} - B_2$ options for its data field, and all together we get

$$A_{M,L}(\tau_{\mathsf{i}},\tau_{\mathsf{d}}) \leqslant q^{L_M}\left(q^{L_M}-B_2\right)^{M-1} = q^{ML_M}\left(1-\frac{B_2}{q^{L_M}}\right)^{M-1}.$$

Lastly, the lower bound on the redundancy is derived to be

$$\begin{aligned} r_{M,L}(\tau_{\mathsf{i}},\tau_{\mathsf{d}}) &= M\cdot L_M - \log_q(A_{M,L}(\tau_{\mathsf{i}},\tau_{\mathsf{d}})) \\ &\geqslant -(M-1)\cdot\log_q\left(1-\frac{B_2}{q^{L_M}}\right) \\ &> \frac{\mathsf{le}_q\cdot(M-1)\cdot B_2}{q^{L_M}}, \end{aligned}$$

where in the last step we used the inequality $-\log_q(1-x) > \mathsf{le}_q\cdot x$ for all $0<x<1$. ∎

As a direct result of Theorem 14, the next corollary holds.

Corollary 15: For $M=q^{\beta L}$ and all τ_{d} such that

$$\tau_{\mathsf{d}} \geqslant L_M\mathcal{H}_q^{-1}\left(\frac{1-2\beta}{1-\beta}+\frac{\log_q(L_M+1)}{(1-\beta)L}\right)+1$$

it holds that $r_{M,L}(\tau_{\mathsf{i}},\tau_{\mathsf{d}})>1$.

Proof: From Theorem 14 it holds that

$$r_{M,L}(\tau_{\mathsf{i}},\tau_{\mathsf{d}}) > \frac{\mathsf{le}_q\cdot(M-1)\cdot B_2}{q^{L_M}}.$$

Hence, $r_{M,L}(\tau_{\mathsf{i}},\tau_{\mathsf{d}})>1$ if $B_2 \geqslant \frac{q^{L_M}}{\mathsf{le}_q\cdot(M-1)}$. According to Lemma 4.8 in [20], $B_2 \geqslant \frac{1}{L_M+1}\cdot q^{L_M\mathcal{H}_q\left(\frac{\tau_{\mathsf{d}}-1}{L_M}\right)}$ and therefore it is enough to show that $\frac{1}{L_M+1}\cdot q^{L_M\mathcal{H}_q\left(\frac{\tau_{\mathsf{d}}-1}{L_M}\right)} \geqslant \frac{q^{L_M}}{\mathsf{le}_q\cdot(M-1)}$. As $M=q^{\beta L}$ and

$$\tau_{\mathsf{d}} \geqslant L_M\mathcal{H}_q^{-1}\left(\frac{1-2\beta}{1-\beta}+\frac{\log_q(L_M+1)}{(1-\beta)L}\right)+1,$$

it holds that

$$\begin{aligned} L_M\mathcal{H}_q\left(\frac{\tau_{\mathsf{d}}-1}{L_M}\right) &\geqslant L_M\cdot\left(\frac{1-2\beta}{1-\beta}+\frac{\log_q(L_M+1)}{(1-\beta)L}\right) \\ &= L_M-\log_q(M)+\log_q(L_M+1) \\ &\geqslant L_M-\log_q(M-1)-\log_q(\mathsf{le}_q)+\log_q(L_M+1). \end{aligned}$$

Finally,

$$\frac{1}{L_M+1}\cdot q^{L_M\mathcal{H}_q\left(\frac{\tau_{\mathsf{d}}-1}{L_M}\right)} \geqslant \frac{q^{L_M}}{\mathsf{le}_q\cdot(M-1)}.$$

∎

Lastly, based upon Corollaries 13 and 15 we conclude with the following result.

Corollary 16: Let $\tau_{\mathsf{d}}{}^*$ be the solution to $r_{M,L}(\tau_{\mathsf{i}},\tau_{\mathsf{d}}{}^*)=1$. For any $\tau_{\mathsf{i}}=o(L)$ and large L it holds that

$$\tau_{\mathsf{d}}{}^* = L_M\cdot\mathcal{H}_q^{-1}\left(\frac{1-2\beta}{1-\beta}\right)(1+o(1))$$

Proof: From Corollary 13 we get that for

$$\tau_{\mathsf{d}} \leqslant L_M\mathcal{H}_q^{-1}\left(\frac{1-2\beta}{1-\beta}-\frac{\log_q(\mathsf{le}_q)}{(1-\beta)L}+\frac{\beta}{1-\beta}\cdot\mathcal{H}_q\left(\frac{\tau_{\mathsf{i}}}{\beta L}\right)\right)$$

it holds that $r_{M,L}(\tau_{\mathsf{i}},\tau_{\mathsf{d}})<1$. Also, by Corollary 15, for

$$\tau_{\mathsf{d}} \geqslant L_M\mathcal{H}_q^{-1}\left(\frac{1-2\beta}{1-\beta}+\frac{\log_q(L_M+1)}{(1-\beta)L}\right)+1$$

is holds that $r_{M,L}(\tau_{\mathsf{i}},\tau_{\mathsf{d}})>1$. When L goes to infinity, we get that both bounds approach

$$L_M\cdot\mathcal{H}_q^{-1}\left(\frac{1-2\beta}{1-\beta}\right)(1+o(1)).$$

Henceforth, the bounds on τ_{d} for $r_{M,L}(\tau_{\mathsf{i}},\tau_{\mathsf{d}})<1$ and $r_{M,L}(\tau_{\mathsf{i}},\tau_{\mathsf{d}})>1$ asymptotically agree and the crossing point $r_{M,L}(\tau_{\mathsf{i}},\tau_{\mathsf{d}})$ lies in between. ∎

An asymptotic improvement to the lower bound on the redundancy from Theorem 14 for $\tau_{\mathsf{i}}=1$, which matches the upper bound from Theorem 12, is proved in the next theorem.

Theorem 17: For fixed $\epsilon,\delta>0$, if $\frac{\tau_{\mathsf{d}}}{L_M} < \mathcal{H}_q^{-1}\left(\frac{1-2\beta}{1-\beta}\right)-\epsilon$, then it holds that

$$A_{M,L}(1,\tau_{\mathsf{d}}) \leqslant q^{ML_M}\left(1-\frac{\log_q(M)\cdot B_2}{q^{L_M}}\right)^{\frac{(q-1)M}{q}}(1+\delta),$$

for L_M large enough, and hence

$$r_{M,L}(1,\tau_{\mathsf{d}}) \geqslant \frac{M(q-1)\cdot\log_q(M)\cdot B_2}{q^{L_M+1}-q\log_q(M)\cdot B_2}-\log(1+\delta).$$

Proof: Let $\mathcal{C}$ be a $(1,\tau_{\mathsf{d}})$-CCC of maximal size $A_{M,L}(1,\tau_{\mathsf{d}})$. First, note that

$$\begin{aligned} C_{\log_q(M)}(2) &= \left|\left\{\boldsymbol{c}\in[q]^{\log_q(M)}\,\middle|\,\sum_{i=0}^{\log_q(M)-1}c_i\equiv 0(\mathrm{mod}\ q)\right\}\right| \\ &= \frac{M}{q}. \end{aligned}$$

The minimum distance of the given set is 2. On the contrary, let $\boldsymbol{c}_1,\boldsymbol{c}_2\in[q]^{\log_q(M)}$ such that $\mathrm{d}_{\mathsf{H}}(\boldsymbol{c}_1,\boldsymbol{c}_2)=1$, and let j be the only index in which $\boldsymbol{c}_{1,j}\neq\boldsymbol{c}_{2,j}$. It holds that $\sum_{i=0}^{\log_q(M)-1}(\boldsymbol{c}_{1,i}-\boldsymbol{c}_{2,i})(\mathrm{mod}\ q)\equiv(\boldsymbol{c}_{1,j}-\boldsymbol{c}_{2,j})\ (\mathrm{mod}\ q)$, which is not $0\ (\mathrm{mod}\ q)$ and therefore it is impossible for them both to belong to the set $C_{\log_q(M)}(2)$. In addition, the size of the set is $\frac{M}{q}$. Let $\boldsymbol{c}\in[q]^{\log_q(M)-1}$, it is trivial that there is a single possible symbol to concatenate into $\boldsymbol{c}$ such that it belongs into the set, therefore there are $q^{\log(M)-1}=\frac{M}{q}$ different words in the set. Also, according to the Singleton bound the size of $C_{\log_q(M)}(2)$ cannot be greater than $\frac{M}{q}$. Let $\mathcal{I}$ be a code of this size. For every set $\mathcal{S}=\{(\mathsf{ind}_0,\boldsymbol{u}_0),\ldots,(\mathsf{ind}_{M-1},\boldsymbol{u}_{M-1})\}\in\mathcal{C}$, let

$$\mathcal{S}_{\text{safe}} = \{\boldsymbol{u}_i|\mathsf{ind}_i\in C\}\in([q]^{L_M})^{M/q}\triangleq\Sigma_{M,L}$$

be the *vector* projection of $\mathcal{S}$ to the data fields of the strands with indices that belong to $\mathcal{I}$ and let

$$I_{\text{safe}}\triangleq\{i\ |\mathsf{ind}_i\in\mathcal{I}\},\ \overline{I}_{\text{safe}}\triangleq[q]^{\log_q(M)}\setminus I_{\text{safe}}.$$

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

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

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

A set of vectors $\boldsymbol{v}=(\boldsymbol{v}_i)_{i\in I_{\text{safe}}}\in\Sigma_{M,L}$ is *good* if for all $i,j\in I_{\text{safe}}$ such that $\mathrm{d}_{\mathsf{H}}(\mathsf{ind}_i,\mathsf{ind}_j)=2$ it holds that $B_{\tau_{\mathsf{d}}-1}(\boldsymbol{v}_i)\cap B_{\tau_{\mathsf{d}}-1}(\boldsymbol{v}_j)=\emptyset$, and otherwise it is *bad*. Denote by $X_{\text{good}},X_{\text{bad}}$ the number of good, bad sets of vectors in $\Sigma_{M,L}$, respectively. If $\boldsymbol{v}\in\Sigma_{M,L}$ is bad, then there are at least two indices $i,j\in I_{\text{safe}}$ such that $\mathrm{d}_{\mathsf{H}}(\mathsf{ind}_i,\mathsf{ind}_j)=2$ and

$B_{\tau_{\mathsf{d}}-1}(\boldsymbol{v}_i)\cap B_{\tau_{\mathsf{d}}-1}(\boldsymbol{v}_j)\neq\emptyset$, i.e., $\mathrm{d}_{\mathsf{H}}(\boldsymbol{v}_i,\boldsymbol{v}_j)\leqslant 2\tau_{\mathsf{d}}-2$. Hence, we get that

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

where $B_3=B_{2\tau_{\mathsf{d}}-2}(L_M)$.

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 with indices in I_{safe}. Since $\boldsymbol{v}$ is a good set of vectors, for every index in $\overline{I}_{\text{safe}}$, the radius-$(\tau_{\mathsf{d}}-1)$ balls of at least $1/(q-1)$ of its neighbor strands are mutually disjoint (have indices in I_{safe}) so there are at most

$$q^{L_M}-\frac{\log_q(M)(q-1)}{q-1}\cdot B_2=q^{L_M}-\log_q(M)\cdot B_2$$

options to choose the data field of the i-th strand. For every bad set of vectors $\boldsymbol{v}\in\Sigma_{M,L}$, it is enough to take the loose bound in which $|\mathcal{C}_{\boldsymbol{v}}|\leqslant\left(q^{L_M}\right)^{\frac{M(q-1)}{q}}$. In conclusion we get that

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

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

$$\left(1-\frac{\log_q(M)\cdot B_2}{q^{L_M}}\right)^{\frac{(q-1)M}{q}}\geqslant q^{-\frac{\mathsf{le}_q\cdot\log_q(M)\cdot B_2(q-1)M}{q^{L_M+1}-\log_q(M)\cdot q\cdot B_2}}.$$

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

$$\begin{aligned}
&\lim_{L_M\to\infty}\frac{\mathsf{le}_q\cdot\log_q(M)\cdot B_2\cdot(q-1)\cdot M}{q^{L_M}-\log_q(M)\cdot q\cdot B_2}\\
&\leqslant\lim_{L_M\to\infty}\frac{\mathsf{le}_q\cdot\log_q(M)\cdot q^{L_M\mathcal{H}_q\left(\frac{\tau_{\mathsf{d}}-1}{L_M}\right)}\cdot(q-1)\cdot M}{q^{L_M+1}-\log_q(M)\cdot q\cdot q^{L_M\mathcal{H}_q\left(\frac{\tau_{\mathsf{d}}-1}{L_M}\right)}}\\
&\leqslant\lim_{L_M\to\infty}\frac{\mathsf{le}_q\cdot\log_q(M)\cdot q^{L_M\mathcal{H}_q\left(\frac{\tau_{\mathsf{d}}-1}{L_M}\right)}\cdot q^{\beta L}}{q^{L_M}}=0,
\end{aligned}$$

and

$$\begin{aligned}
&\lim_{L_M\to\infty}\frac{M(\log_q(M)\cdot(q-1))^2\cdot B_3}{q^{L_M}}\\
&\leqslant\lim_{L_M\to\infty}\frac{M(\log_q(M)\cdot(q-1))^2\cdot q^{L_M\mathcal{H}\left(\frac{2(\tau_{\mathsf{d}}-1)}{L_M}\right)}}{q^{L_M}}\\
&\leqslant\lim_{L\to\infty}\frac{q^{\beta L}\cdot(\beta L)^2\cdot q^{(1-2\beta-\epsilon')L}}{q^{(1-\beta)L}}=\lim_{L\to\infty}\frac{(\beta L)^2}{q^{\epsilon' L}}=0,
\end{aligned}$$

for some $\epsilon'>0$. Thus,

$$\lim_{L\to\infty}\frac{M(\log_q(M)(q-1))^2\cdot B_3}{q^{L_M}\cdot\left(1-\frac{\log_q(M)\cdot B_2}{q^{L_M}}\right)^{\frac{(q-1)M}{q}}}=0,$$

which confirms the theorem's statements. ■

V. A Construction of CCCs

In this section we propose a construction of CCCs. It is shown that with a single symbol of redundancy it is possible to construct CCCs for relatively large values of τ_{d}.

The algorithm will use the following functions (For a detailed demonstration of these functions refer to Example 18):

- The function $w_\ell(S,t)$ is defined over a set of vectors S and a positive integer t and outputs a vector $\boldsymbol{w}\in[q]^\ell$ which satisfies the following condition. For all $\boldsymbol{v}\in S$, $\mathrm{d}_{\mathsf{H}}(\boldsymbol{w},\boldsymbol{v}_{[\log_q(M),\ell]})\geqslant t$. The value of ℓ will be determined later as a function of τ_{i}, τ_{d}, 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 τ_{i} using $\tau_{\mathsf{i}}\lceil\log_q(\log_q(M)\cdot(q-1))\rceil$ symbols which mark the positions where the indices $\mathsf{ind}_i,\mathsf{ind}_j$ differ and the difference between the values in these positions. It is possible that there are less than τ_{i} such indices. In such a case the last index is replicated to get the desired length. All indices are written in an increasing order
- The function $\Delta_2(\boldsymbol{u}_i,\boldsymbol{u}_j)$ encodes the difference between the two data fields $\boldsymbol{u}_i,\boldsymbol{u}_j\in[q]^{L_M}$ of Hamming distance at most $\tau_{\mathsf{d}}-1$ using $(\tau_{\mathsf{d}}-1)\lceil\log_q(L_M\cdot(q-1))\rceil$ symbols which mark the positions where they differ and the difference between the values in these positions. It is possible that there are less than $\tau_{\mathsf{d}}-1$ such indices. In such a case the last index is replicated to get the desired length. All indices are written in an increasing order and in case $\boldsymbol{u}_i=\boldsymbol{u}_j$, $\Delta_2(\boldsymbol{u}_i,\boldsymbol{u}_j)$ will output a sequence of indices in a decreasing order in order to flag this case.

Note that the functions $\Delta_1(\mathsf{ind}_i,\mathsf{ind}_j)$, $\Delta_2(\boldsymbol{u}_i,\boldsymbol{u}_j)$ can be optimized to use $\lceil\log_q(B_{\tau_{\mathsf{i}}}(\log_q(M)))\rceil$, $\lceil\log_q(B_{\tau_{\mathsf{d}}-1}(L_M))\rceil$ symbols respectively. We will consider both options and use either one of them based upon the statement we claim.

The input to the algorithm is a set of M vectors $\boldsymbol{v}_0,\ldots,\boldsymbol{v}_{M-1}$. All vectors are of length L_M symbols, except for $\boldsymbol{v}_{M-1}$ which has length of L_M-1 symbols. 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. One symbol is added to $\boldsymbol{v}_{M-1}$, hence, the code has a single symbol of redundancy, to mark whether some vectors were altered by the algorithm.

For a given word $\mathcal{S}=\{(\mathsf{ind}_0,\boldsymbol{u}_0),\ldots,(\mathsf{ind}_{M-1},\boldsymbol{u}_{M-1})\}$, the notation $S(\tau_{\mathsf{i}},i)$ in the algorithm, for $i\in[M]$, will be used as a shortcut to the set $\{\boldsymbol{u}_j\mid(\mathsf{ind}_j,\boldsymbol{u}_j)\in\mathcal{S}\,,\mathrm{d}_{\mathsf{H}}(\mathsf{ind}_i,\mathsf{ind}_j)\leqslant\tau_{\mathsf{i}},j\neq i\}$ of data fields corresponding to indices ind_j of Hamming distance at most τ_{i} from ind_i. The elements in $S(\tau_{\mathsf{i}},i)$ are referred as the neighbors of $\boldsymbol{u}_i$ and $S(\tau_{\mathsf{i}},i)$ itself as its neighborhood. At any iteration of the while loop, when the i-th strand is corrected, the function $w_\ell(S(\tau_{\mathsf{i}},i),\tau_{\mathsf{d}})$ will be

Algorithm 3 $(\tau_{\mathsf{i}}, \tau_{\mathsf{d}}) - CCC$ Construction

Input: M vectors $\boldsymbol{v}_0, \ldots, \boldsymbol{v}_{M-1}$ such that $\boldsymbol{v}_0, \ldots, \boldsymbol{v}_{M-2} \in [q]^{L_M}$ and $\boldsymbol{v}_{M-1} \in [q]^{L_M - 1}$

Output: a codeword $\mathcal{S} = \{(\mathsf{ind}_0, \boldsymbol{u}_0), \ldots, (\mathsf{ind}_{M-1}, \boldsymbol{u}_{M-1})\}$

1: $\forall i \in [M-1] : \boldsymbol{u}_i = \boldsymbol{v}_i, \boldsymbol{u}_{M-1} = (\boldsymbol{v}_{M-1}, 0)$
2: $p \leftarrow M - 1$
3: $B \leftarrow \{(i,j) \mid i < j \wedge \mathrm{d}_{\mathsf{H}}(\mathsf{ind}_i, \mathsf{ind}_j) \leqslant \tau_{\mathsf{i}} \wedge \mathrm{d}_{\mathsf{H}}(\boldsymbol{u}_i, \boldsymbol{u}_j) < \tau_{\mathsf{d}}\}$
4: **while** $B \neq \emptyset$ **do**
5: Pick $(i,j) \in B$
6: $\mathsf{repl} \leftarrow (w_\ell(S(\tau_{\mathsf{i}}, i), \tau_{\mathsf{d}}), \Delta_1(\mathsf{ind}_i, \mathsf{ind}_j), \Delta_2(\boldsymbol{u}_i, \boldsymbol{u}_j))$
7: $(\boldsymbol{u}_p)_{L_M - 1} \leftarrow 1$
8: $(\boldsymbol{u}_p)_{[0, \log_q(M)]} \leftarrow \mathsf{ind}_i$
9: $(\boldsymbol{u}_i)_{[\log_q(M), \mathrm{len}]} \leftarrow \mathsf{repl}$
10: $B \leftarrow \{(i,j) \mid i < j \wedge \mathrm{d}_{\mathsf{H}}(\mathsf{ind}_i, \mathsf{ind}_j) \leqslant \tau_{\mathsf{i}} \wedge \mathrm{d}_{\mathsf{H}}(\boldsymbol{u}_i, \boldsymbol{u}_j) < \tau_{\mathsf{d}}\}$
11: $p \leftarrow i$
12: **end while**
13: $(\boldsymbol{u}_p)_{L_M - 1} \leftarrow 0$
14: $(\boldsymbol{u}_p)_{[0, \log_q(M)]} \leftarrow (\boldsymbol{v}_{M-1})_{[0, \log_q(M)]}$

	index	data
$(\mathsf{ind}_0, \boldsymbol{v}_0) =$	**000**	011011100110000101101110011000001
$(\mathsf{ind}_1, \boldsymbol{v}_1) =$	**001**	011011100110000101101110011000001
$(\mathsf{ind}_2, \boldsymbol{v}_2) =$	**010**	001000000110111001100000101101110
$(\mathsf{ind}_3, \boldsymbol{v}_3) =$	**011**	011000010110111001100000101101110
$(\mathsf{ind}_4, \boldsymbol{v}_4) =$	**100**	011000010010000001101110011000001
$(\mathsf{ind}_5, \boldsymbol{v}_5) =$	**101**	011011100110000101101110011000001
$(\mathsf{ind}_6, \boldsymbol{v}_6) =$	**110**	011011100110000100100000001101110
$(\mathsf{ind}_7, \boldsymbol{v}_7) =$	**111**	011000010110111001100000100000000?

Fig. 4. The input data for Algorithm 3 in Example 18. The red question mark indicates the redundancy bit that has not been set yet.

used to update the data field $\boldsymbol{u}_i$ such that it does not violate the clustering constraint and yet can be decoded. In order to make room for the vector generated by the function $w_\ell(S(\tau_{\mathsf{i}}, i), \tau_{\mathsf{d}})$, $\boldsymbol{u}_i$ is encoded based on its similarity to one of its close neighbors $\boldsymbol{u}_j$. These modifications are encoded together as a repelling vector of length

$$\mathrm{len} = \ell + \lceil \log_q(B_{\tau_{\mathsf{i}}}(\log_q(M))) \rceil + \lceil \log_q(B_{\tau_{\mathsf{d}}-1}(L_M)) \rceil.$$

Finally the vectors that were altered are chained, starting at $\boldsymbol{u}_{M-1}$, so it is possible to traverse these vectors and restore the original data.

Example 18: We demonstrate the encoding process by applying Algorithm 3 for $\tau_{\mathsf{i}} = 1, \tau_{\mathsf{d}} = 4$ on input of $M = 8$ strands over the binary alphabet. Each strand consists of an index field of length 3 and a data field of length 28, hence, $L = 31$. We will also use $w_\ell(S, i)$ for $\ell = 8$, using the construction that will be described in Lemma 21.

There are eight strands over the binary alphabet as can be seen in Fig. 4. These strands are encoded such that the resulting codeword satisfies the $(1,4)$-clustering constraint.

$\mathrm{d}_{\mathsf{H}}(\boldsymbol{v}_0, \boldsymbol{v}_1) = 0$ $\mathrm{d}_{\mathsf{H}}(\boldsymbol{v}_1, \boldsymbol{v}_5) = 0$ $\mathrm{d}_{\mathsf{H}}(\boldsymbol{v}_4, \boldsymbol{v}_5) = 6$
$\mathrm{d}_{\mathsf{H}}(\boldsymbol{v}_0, \boldsymbol{v}_2) = 16$ $\mathrm{d}_{\mathsf{H}}(\boldsymbol{v}_2, \boldsymbol{v}_3) = 2$ $\mathrm{d}_{\mathsf{H}}(\boldsymbol{v}_4, \boldsymbol{v}_6) = 14$
$\mathrm{d}_{\mathsf{H}}(\boldsymbol{v}_0, \boldsymbol{v}_4) = 6$ $\mathrm{d}_{\mathsf{H}}(\boldsymbol{v}_2, \boldsymbol{v}_6) = 10$ $\mathrm{d}_{\mathsf{H}}(\boldsymbol{v}_5, \boldsymbol{v}_7) \geqslant 15$
$\mathrm{d}_{\mathsf{H}}(\boldsymbol{v}_1, \boldsymbol{v}_3) = 16$ $\mathrm{d}_{\mathsf{H}}(\boldsymbol{v}_3, \boldsymbol{v}_7) \geqslant 5$ $\mathrm{d}_{\mathsf{H}}(\boldsymbol{v}_6, \boldsymbol{v}_7) \geqslant 15$

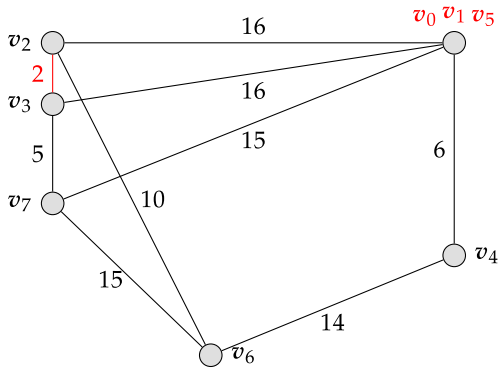

Fig. 5. Distances of the data fields visualization for all pairs of strands with Hamming distance of 1 between their index fields. Pairs violating the $(1,4)-$clustering constraint are colored in red. Note that v_0, v_1, and v_5 are in the same node since their data is the same.

The first step is to compute the set B of pairs that violate this constraint in Step 3. As can be seen in Fig. 5, $B = \{(1,5), (2,3), (0,1)\}$. Those are the pairs of indices with Hamming distance 1 for which the data fields of the strands have Hamming distance less than 4.

On the first iteration of the while loop in Step 4 we handle the pair $(1,5)$. First we set the flag bit in $\boldsymbol{u}_7$ to 1 (Step 7) and we also need to update $\boldsymbol{u}_7$ to link to $\boldsymbol{u}_1$ in the chain (Step 8). The data fields $\boldsymbol{u}_1$ and $\boldsymbol{u}_5$ are identical, hence, $\Delta_2(\boldsymbol{u}_1, \boldsymbol{u}_5)$ can be any unsorted sequence of indices, for example $25, 16, 22$ or $(11001\ 10000\ 10110)$. Also $\Delta_1(\mathsf{ind}_1, \mathsf{ind}_5) = (11)$ as those differ in the third bit. For $w_\ell(S(\tau_{\mathsf{i}}, i), \tau_{\mathsf{d}})$ we take the vector (10011100). After calculating the repelling terms $w_8(S(1,1))$, $\Delta_1(\mathsf{ind}_1, \mathsf{ind}_5), \Delta_2(\boldsymbol{u}_1, \boldsymbol{u}_5)$ we have the whole repelling vector

$$\mathsf{repl} = (10011100\ 11\ 11001\ 10000\ 10110),$$

and $\boldsymbol{u}_1$ is updated as in Step 9. Note that after altering $\boldsymbol{u}_1$, it satisfies the clustering constraint with respect to both $\boldsymbol{u}_0$ and $\boldsymbol{u}_5$. Therefore, after updating the set B (Step 10) it contains a single pair $(2,3)$.

Moving on to the next iteration we handle the pair $(2,3)$. We start by placing a link to $\boldsymbol{u}_2$ in $\boldsymbol{u}_1$ (Step 8). We set the flag bit of $\boldsymbol{u}_1$ to be 1 as well (Step 7). Now $\Delta_1(\mathsf{ind}_2, \mathsf{ind}_3) = (01)$. For $w_\ell(S(\tau_{\mathsf{i}}, i), \tau_{\mathsf{d}})$ we take the vector (10011100) based on Lemma 21 as before. The two data fields differ in the indices 1 and 7, therfore $\Delta_2(\boldsymbol{u}_2, \boldsymbol{u}_3) = (00001\ 00111\ 00111)$. Updating B again (Step 10) results with an empty set. Hence, the flag bit of $\boldsymbol{u}_2$ is set to 0 (Step 13) and its first $\log_2(M) = 3$ bits are set to $v_{7[0,3]} = (011)$ (Step 14). This ends the process and the result can be seen in Fig. 6.

Theorem 19: For any input vectors $\boldsymbol{v}_0, \ldots, \boldsymbol{v}_{M-1}$, Algorithm 3 returns a valid $(\tau_{\mathsf{i}}, \tau_{\mathsf{d}})$-CCC codeword for any τ_{d} that

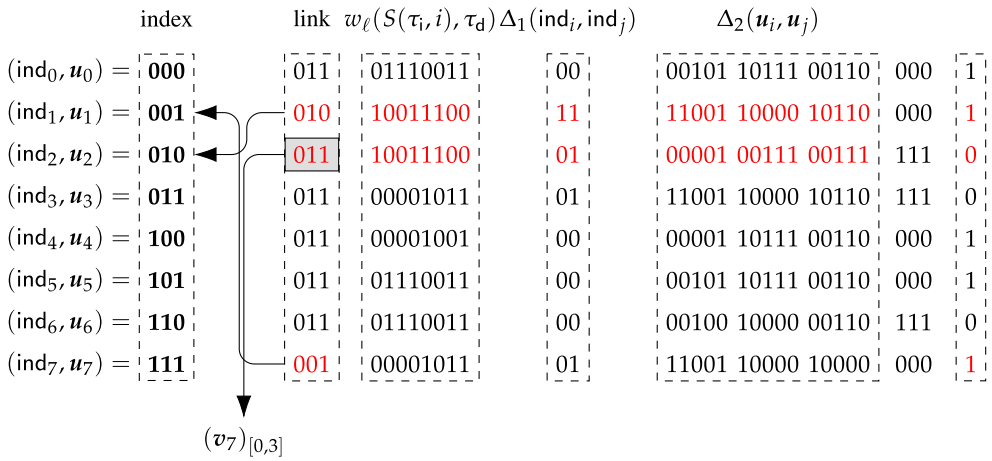

Fig. 6. The output of Algorithm 3 on the data given in Example 18. Bits changed by the algorithm are colored in red.

satisfies:

$$L-2\log_q(M) \geqslant \ell+\log_q(B_{\tau_{\mathsf{i}}}(\log_q(M))\cdot B_{\tau_{\mathsf{d}}-1}(L_M))+3.$$

Furthermore, it is possible to decode the vectors $\boldsymbol{v}_0,\ldots,\boldsymbol{v}_{M-1}$, and Algorithm 3 uses a single symbol of redundancy.

Proof: Algorithm 3 starts by initializing the data fields $\boldsymbol{u}_0,\ldots,\boldsymbol{u}_{M-1}$ of the output set $\mathcal{S}$, with the input vectors $\boldsymbol{v}_0,\ldots,\boldsymbol{v}_{M-1}$, while adding the zero symbol at the end of the data field $\boldsymbol{u}_{M-1}$ to mark that no operation is needed for the decoding at this point. 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.

Each strand modified by the algorithm has its index written as part of the data field of the strand modified in the previous iteration of the while loop. Thus, the algorithm creates a linked list of the modified strands. This list will be used in the decoding process and will be referred as the *decoding chain*, or simply the *chain*.

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 symbol at the end of the previous strand is changed to 1 (Step 7). This denotes that it is not the last strand in the decoding chain. In Step 6, the algorithm calculates the vector $\boldsymbol{w}=w_\ell(S(\tau_{\mathsf{i}},i),\tau_{\mathsf{d}})$ and embeds it in the data field of the i-th strand in Step 9.

In Steps 7 and 8 $\boldsymbol{u}_p$ is modified. In the first iteration if $j=p=M-1$, it is possible that $\mathrm{d}_{\mathsf{H}}(\boldsymbol{u}_i,\boldsymbol{u}_j)$ is affected by the modifications in $\boldsymbol{u}_p$ and is greater than or equal to τ_{d}. For this reason the computation of $\Delta_2(\boldsymbol{u}_i,\boldsymbol{u}_j)$ takes place before the modifications of $\boldsymbol{u}_i$. Denote by $(\boldsymbol{u}_i)^*$ the value of $\boldsymbol{u}_i$ after it was changed. The vector $\boldsymbol{w}$ satisfies that for all $\boldsymbol{u}\in S(\tau_{\mathsf{i}},i)$, $\mathrm{d}_{\mathsf{H}}\left(\boldsymbol{w},\boldsymbol{u}_{[\log_q(M),\ell]}\right)\geqslant\tau_{\mathsf{d}}$. Since after Step 9 $(\boldsymbol{u}_i)^*_{[\log_q(M),\ell]}=\boldsymbol{w}$ it is deduced that for all $\boldsymbol{u}\in S(\tau_{\mathsf{i}},i)$,

$$\mathrm{d}_{\mathsf{H}}\left((\boldsymbol{u}_i)^*_{[\log_q(M),\ell]},\boldsymbol{u}_{[\log_q(M),\ell]}\right)\geqslant\tau_{\mathsf{d}}$$

and thus $\mathrm{d}_{\mathsf{H}}\left((\boldsymbol{u}_i)^*,\boldsymbol{u}\right)\geqslant\tau_{\mathsf{d}}$. 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 in this iteration of the while loop. 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, no bad pairs with the index i have been created. As mentioned, in Steps 7 and 8 $\boldsymbol{u}_p$ is modified. In general, when $\boldsymbol{u}_p$ is modified in these steps, this does not create any new entries in B, since $[\boldsymbol{u}_p]_{[\log_q(M),len]}$ is not altered and this field has been chosen in the previous step such that the distance to all of its neighbors is large enough. Yet, this does not hold in the first iteration when $p=M-1$. Hence, the size of B can increase on the first iteration. That is, the size of the set B decreases on each iteration except to the first one, and the algorithm terminates. The constraint

$$\begin{aligned}L-2\log_q(M) &\geqslant \ell+\log_q(B_{\tau_{\mathsf{i}}}(\log_q(M))\cdot B_{\tau_{\mathsf{d}}-1}(L_M))+3\\ &\geqslant \ell+\lceil\log_q(B_{\tau_{\mathsf{i}}}(\log_q(M)))\rceil+\lceil\log_q(B_{\tau_{\mathsf{d}}-1}(L_M))\rceil+1,\end{aligned}$$

guarantees that the data field is large enough in order to write the information required on each update step of the while loop.

In Step 8 the first $\log_p(M)$ bits of $\boldsymbol{u}_p$ are overwritten with ind_i which is the index of the strand that is about to be modified on this iteration. Also, in Step 11, p is updated to have the value of i. Thus, the algorithm creates a chain starting at $\boldsymbol{u}_{M-1}$, that can be traversed by reading these $\log_p(M)$ bits of the link and interpreting them as the index of the next strand in the chain. The strand $\boldsymbol{u}_{M-1}$ cannot be altered, because it results in loss of parts of the chain. To make sure it will not happen, the data field of the strand with the smaller index in the pair is always the one to be altered. Additional $\log_q(M)$ symbols are required to encode the index of the first strand in the chain. For this purpose, the first $\log_q(M)$ symbols of $\boldsymbol{v}_{M-1}$ are placed at the end of the chain (Step 14). This is possible because the last strand in the chain has $\log_q(M)$

spare symbols, as there is no need to encode the index for the next strand.

The idea of the decoding process is to track the chain of the strands and then traverse the chain in the opposite direction while recovering the input vectors. The decoding process starts with the $(M-1)$-st strand. If the flag at the end of the data field $\boldsymbol{u}_{M-1}$ is zero then there is nothing to be done. Otherwise, the first $\log_q(M)$ symbols of $\boldsymbol{u}_{M-1}$ indicate the index ind_i of the first altered vector $\boldsymbol{u}_i$. This process repeats until we encounter a vector with a flag symbol of value zero. This is the last vector in the chain, and furthermore, its first $\log_q(M)$ symbols of the data field are the first $\log_q(M)$ symbols of $\boldsymbol{v}_{M-1}$. We traverse the chain in the opposite direction, recovering each vector using the vectors $\Delta_1(\mathsf{ind}_i, \mathsf{ind}_j), \Delta_2(\boldsymbol{u}_i, \boldsymbol{u}_j)$ that are encoded in the data field of the i-th strand. We stop at $\boldsymbol{u}_{M-1}$. Then, we recover $\boldsymbol{v}_{M-1}$ which requires only to place back its original $\log_q(M)$ symbols. Note that this is sufficient to recover $\boldsymbol{v}_{M-1}$ as the $(M-1)$-th strand will not be changed after the first iteration where $\boldsymbol{u}_p = \boldsymbol{u}_{M-1}$.

This decoding process works as the order we go through the chain in the second time is exactly the inverse of the encoding order. This way all of the encoding operations can be reverted, just as running the algorithm backwards. Note that the decoding is done backwards as each of the altered vectors is encoded with respect to some neighbor that may be changed later on in the encoding process. Since the decoding is operated in a reverse order, this guarantees that each vector is recovered with respect to the exact same vector in which it was encoded. Recall that in the first iteration there is the possibility that $\boldsymbol{u}_j = \boldsymbol{u}_p = \boldsymbol{u}_{M-1}$. In this case, it will not be possible to decode the first strand that was altered without first recovering $\boldsymbol{u}_{M-1}$. Fortunately, the only $\log_p(M)$ bits that were altered in $\boldsymbol{u}_{M-1}$ are stored at the end of the chain. Hence, the decoding process yields the correct original data for all strands. ∎

Remark 4: Algorithm 3 achieves a single bit of redundancy using the fact that two strands with similar data can be compressed and represented with a relatively small number of symbols. This approach is used in other works as well. For example in [25], this method was used in encoding algorithms for constraints over two-dimensional binary arrays. More specifically, one of the array constraints considered in that paper, referred by t-conservative arrays, imposes each row and each column to have at least t transitions of the form '0' $\rightarrow$ '1' or '1' $\rightarrow$ '0'. Rows that have less than t such transitions are called *t-bad* rows, and the same applies for *t-bad* columns. Those rows and columns can be 'recycled' during the encoding process, taking advantage of the fact that those rows/columns can be represented using a relatively small number of symbols. The spare space is used to encode the information required to decode the original rows and columns. In particular they maintain a linked list used in the decoding to get which rows and columns were recycled.

Theorem 20: The complexity of Algorithm 3 is $O(M \cdot B_{\tau_{\mathsf{i}}}(\log_q(M)) \cdot B_{\tau_{\mathsf{d}}-1}(\ell) \cdot L + M \cdot L^2)$ symbol operations. Furthermore, the complexity of the decoding process is $O(M \cdot L^2)$ symbol operations.

Proof: For the encoding, we first note that each vector $\boldsymbol{u}_i$ is modified at most once. This is due to the fact that after the modifying any of the strands, it satisfies the constraint with respect to any of its neighbors. Hence, it is enough to go through the strands one by one, and check if any modification is needed. In other words, it is checked whether this strand violates the constraint with respect to some other strand. Checking every strand requires a comparison with $B_{\tau_{\mathsf{i}}}(\log_q(M))$ other strands. In case the strand has to be modified, compressing it, i.e. computing $\Delta_1(\mathsf{ind}_i, \mathsf{ind}_j)$, $\Delta_($\boldsymbol{u}_i, \boldsymbol{u}_j)2, requires $O(L^2)$ symbol operations. This is done by computing the index of the compressed data ind_i, $\boldsymbol{u}_i$ in the ball $B_{\tau_{\mathsf{i}}}(\mathsf{ind}_j)$,$B_{\tau_{\mathsf{d}}}(\boldsymbol{u}_j)$ respectively, based on the method presented in Proposition 1 of the enumerative coding scheme in [7]. This method requires the computation of a sum with $q \cdot L$ elements, where each of them is a binomial coefficient. The binomial coefficients $\binom{n}{k}$ can be pre-calculated for all $n, k \leqslant L$, yielding a quadratic complexity in L. Lastly, finding the vector $w_\ell(S(\tau_{\mathsf{i}}, i), \tau_{\mathsf{d}})$ can be done with a brute force search that is guaranteed to end after at most $B_{\tau_{\mathsf{i}}}(\log_q(M)) \cdot B_{\tau_{\mathsf{d}}-1}(\ell)$ iterations, each requires ℓ symbol operations. Therefore, modifying every strand at each iteration of the while loop requires at most

$$O(B_{\tau_{\mathsf{i}}}(\log_q(M)) \cdot L + L^2 + B_{\tau_{\mathsf{i}}}(\log_q(M)) \cdot B_{\tau_{\mathsf{d}}-1}(\ell) \cdot \ell)$$

symbol operations, which is equivalent to

$$O(B_{\tau_{\mathsf{i}}}(\log_q(M)) \cdot B_{\tau_{\mathsf{d}}-1}(\ell) \cdot L + L^2)$$

symbol operations. Over all M strands, the overall complexity of the encoding process is

$$O(M \cdot B_{\tau_{\mathsf{i}}}(\log_q(M)) \cdot B_{\tau_{\mathsf{d}}-1}(\ell) \cdot L + M \cdot L^2).$$

For the decoding, we need to traverse the chain starting at $\boldsymbol{u}_{M-1}$. This requires reading at most M indices, or $M \log_q(M)$ symbol operations. Next, going backwards in the chain, we need to extract the compressed data. That is, for some $\boldsymbol{u}_i$ that is compressed with respect to $\boldsymbol{u}_j$, we need to read the values of $\Delta_1(\mathsf{ind}_i, \mathsf{ind}_j)$, $\Delta_2(\boldsymbol{u}_i, \boldsymbol{u}_j)$, followed by reading $\boldsymbol{u}_j$ and computing the original data stored in $\boldsymbol{u}_i$. This is done by performing the reverse function of the enumerative coding scheme presented in [7]. In order to recover one symbol at a time. This process also requires $O(L^2)$ symbol operations. Overall each strand requires

$$\lceil \log_q(B_{\tau_{\mathsf{i}}}(\log_q(M))) \rceil + \lceil \log_q(B_{\tau_{\mathsf{d}}-1}(L_M)) \rceil + L^2 + L \cdot q$$

symbol operations which is $O(L^2)$ symbol operations for extraction. Therefore, the overall complexity of the decoding process is $O(M \cdot L^2)$. ∎

Next we discuss the function $w_\ell(S, t)$. This function takes a set of vectors S as input and outputs a vector $\boldsymbol{w} \in [q]^\ell$ such that for all $\boldsymbol{v} \in S_{[\log_q(M),\ell]} \triangleq \{\boldsymbol{v}_{[\log_q(M),\ell]} \mid \boldsymbol{v} \in S\}$, it holds that $\mathrm{d}_{\mathsf{H}}(\boldsymbol{w}, \boldsymbol{v}) \geqslant t$. The length ℓ of the vector $\boldsymbol{w}$ is determined by the smallest value of ℓ for which

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

That is, a 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 $S_{[\log_q(M),\ell]}$.

We first show how such a vector $w_\ell(S, t)$ can be constructed efficiently. The presented method is not optimal in the output's

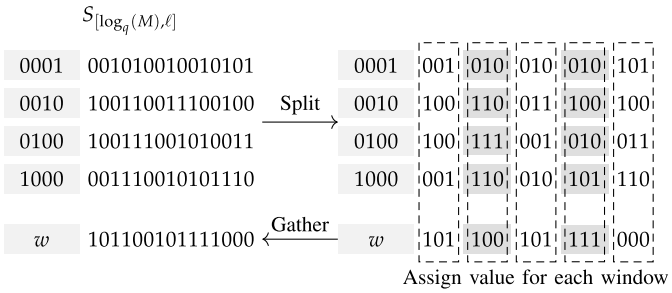

Fig. 7. Construction of $w_\ell(S(e\tau_{\mathsf{i}},0),t)$ for $M=16$, $\ell=12$, $\tau_{\mathsf{i}}=1$, and $t=5$.

length. However, it does improve the encoding complexity with respect to the one presented in Theorem 20, which requires an exhaustive search.

Lemma 21: For all t, a vector $w_\ell(S,t)$ can be constructed for $\ell = t\cdot\lceil\log_q(|S|+1)\rceil$. The complexity of the construction process is $O(t\cdot|S|\cdot\log_q(|S|))$ symbol operations.

Proof: Denote the required output by $\boldsymbol{w}$. In order to construct $\boldsymbol{w}$ as required, we split each $v\in S_{[\log_q(M),\ell]}$ into t windows, each of length $\lceil\log_q(|S|+1)\rceil$. Note that for each such window we can choose at least $q^{\log_q(|S|+1)}=|S|+1$ different values. On the other hand, for the i-th window, we have at most $|S|$ different values in $v\in S_{[\log_q(M),\ell]}$. That is, we can find a value that does not appear in this window, and hence it has Hamming distance at least 1 from any other value in this window. Concatenating the values of the t different windows we get a vector $\boldsymbol{w}$ that has distance at least t from each of the vectors in $S_{[\log_q(M),\ell]}$.

For the complexity, it is clear that each window can be handled independently. For each of these windows we use a binary vector of size $|S|+1$ that has entry for each possible value over $\log_q(|S|+1)$ symbols. The binary vector is initiated in a way that it has 1 in some entry if its index representation over $[q]$ appears in the window. Then, we only need to find an entry in the binary vector with value 0. Searching the binary vector requires at most $|S|+1$ bit operations. Initiating the binary vector requires $|S|\cdot\log_q(|S|+1)$ symbol operations. We get an overall of

$$(|S|+1)\cdot\log_q(|S|+1)+(|S|+1)=O(|S|\cdot\log_q(|S|))$$

symbol operations. And for all windows together,

$$O(t\cdot|S|\cdot\log_q(|S|)).$$

■

Example 22: We demonstrate the computation of $w_\ell(S,t)$ using the construction from Lemma 21. In this example $S=S(1,1)$ and $t=4$ for the same strands in Example 18.

Recall that in Example 18 we picked $\boldsymbol{u}_1$ as the first strand to be fixed. In order to fix $\boldsymbol{u}_1$ we had to compute $w_\ell(S(1,1),4)$ for $\ell=4\cdot\lceil\log_2(|S(1,1)|+1)\rceil=4\cdot(2+1)=8$. Note that,

$$S(1,1)_{[3,8]}=\{(01110011),(00001011),(01110011)\},$$

and hence, we seek to find a vector $\boldsymbol{w}$ such that $\mathrm{d}_{\mathsf{H}}(\boldsymbol{w},01110011)\geqslant 4$ and also $\mathrm{d}_{\mathsf{H}}(\boldsymbol{w},00001011)\geqslant 4$. using the construction from Lemma 21, $S(1,1)_{[3,8]}$ is split into $t=4$ windows:

$$S(1,1)_{[3,2]}=\{(00),(01)\},S(1,1)_{[5,2]}=\{(00),(11)\},$$
$$S(1,1)_{[7,2]}=\{(00),(10)\},S(1,1)_{[9,2]}=\{(11)\}.$$

For every window we find $\boldsymbol{w}_i$ such that $\boldsymbol{w}_i\notin S(1,1)_{[3+2i,2]}$, and get $\boldsymbol{w}=(\boldsymbol{w}_0,\boldsymbol{w}_1,\boldsymbol{w}_2,\boldsymbol{w}_3)$. For example a possible choice of the vector is

$$\boldsymbol{w}_0=(10)\notin S(1,1)_{[3,2]},\boldsymbol{w}_1=(01)\notin S(1,1)_{[5,2]},$$
$$\boldsymbol{w}_2=(11)\notin S(1,1)_{[7,2]},\boldsymbol{w}_3=(00)\notin S(1,1)_{[9,2]}.$$

Then $w_\ell(S(1,1),4)=\boldsymbol{w}=(10011100)$ and it holds that

$$\mathrm{d}_{\mathsf{H}}(\boldsymbol{w},01110011)\geqslant 4,\mathrm{d}_{\mathsf{H}}(\boldsymbol{w},00001011)\geqslant 4.$$

In Algorithm 3, the function w_ℓ is evaluated for $w_\ell(S(\tau_{\mathsf{i}},i),\tau_{\mathsf{d}})$, and hence $S=S(\tau_{\mathsf{i}},i)$. The size of the set $S(\tau_{\mathsf{i}},i)$ is at most $B_{\tau_{\mathsf{i}}}(\log_q(M))-1$, and therefore, $|S_{[\log_q(M),\ell]}|\leqslant B_{\tau_{\mathsf{i}}}(\log_q(M))-1$. We denote by $\ell(\tau_{\mathsf{i}},\tau_{\mathsf{d}},M)$ the smallest value of ℓ such that

$$q^\ell>\big(B_{\tau_{\mathsf{i}}}(\log_q(M))-1\big)\cdot B_{\tau_{\mathsf{d}}-1}(\ell).$$

Next, we study the value of $\ell(\tau_{\mathsf{i}},\tau_{\mathsf{d}},M)$. From Lemma 21 we derive a construction for $w_\ell(S,t)$ of length $t\cdot\lceil\log_q(|S|+1)\rceil$. Hence, for $S=S(\tau_{\mathsf{i}},i)$ it holds that

$$\ell(\tau_{\mathsf{i}},\tau_{\mathsf{d}},M)\leqslant\tau_{\mathsf{d}}\cdot\lceil\log_q(B_{\tau_{\mathsf{i}}}(\log_q(M)))\rceil.$$

The following upper bound $B_{\tau_{\mathsf{i}}}(\log_q(M))\leqslant\big(\log_q(M)\cdot(q-1)\big)^{\tau_{\mathsf{i}}}$ also suggests that

$$\ell(\tau_{\mathsf{i}},\tau_{\mathsf{d}},M)\leqslant\tau_{\mathsf{d}}\cdot\big(\tau_{\mathsf{i}}\cdot\log_q\log_q(M)\ +\tau_{\mathsf{i}}\cdot\log_q(q-1)+1\big).$$

The next lemma provides a better upper bound on the value of $\ell(\tau_{\mathsf{i}},\tau_{\mathsf{d}},M)$.

The next lemma will be used in the subsequent one and in the rest of the paper. Its proof is deferred to Appendix A.

Lemma 23: Let $x\in[0,1]$. It holds that

$$4\log_q(2)\cdot x(1-x)\leqslant\mathcal{H}_q(x)\leqslant 2\log_q(2)\sqrt{x(1-x)}+x.$$

Lemma 24: For all $\tau_{\mathsf{i}},\tau_{\mathsf{d}},M$ it holds that

$$\ell(\tau_{\mathsf{i}},\tau_{\mathsf{d}},M)\leqslant 5\tau_{\mathsf{d}}+2\tau_{\mathsf{i}}\cdot\log_q\log_q(M)\ +2\tau_{\mathsf{i}}\cdot\log_q(q-1).$$

Proof: We start by denoting the size of $S_{[\log_q(M),\ell]}$ by N. We show that for $\ell\geqslant 5\tau_{\mathsf{d}}+2\log_q(N)-5$ the following inequality is satisfied:

$$q^\ell>N\cdot B_{\tau_{\mathsf{d}}-1}(\ell),$$

and therefore for $N=B_{\tau_{\mathsf{i}}}(\log_q(M))-1$, using the inequality $B_{\tau_{\mathsf{i}}}(\log_q(M))\leqslant\big(\log_q(M)\cdot(q-1)\big)^{\tau_{\mathsf{i}}}$, we achieve the desired result:

$$\begin{aligned}\ell(\tau_{\mathsf{i}},\tau_{\mathsf{d}},M)&\leqslant 5\tau_{\mathsf{d}}+2\log_q(N)-5\\&\leqslant 5\tau_{\mathsf{d}}+2\tau_{\mathsf{i}}\cdot\log_q\log_q(M)\ +2\tau_{\mathsf{i}}\cdot\log_q(q-1).\end{aligned}$$

Let $\ell\geqslant 5\tau_{\mathsf{d}}+2\log_q(N)-5$ and $v=\tau_{\mathsf{d}}-1$. Using the inequality of arithmetic and geometric means, $2\sqrt{a\cdot b}\leqslant a+b$ (where equality holds if and only if $a=b$) it holds that

$$\ell\geqslant 5v+2\log_q(N)>3v+\log_q(N)+2\sqrt{v\cdot\big(v+\log_q(N)\big)}$$

with $a = v$, $b = v + \log_q(N)$ and note that $a \neq b$. Rearranging this inequality and squaring both sides we get that

$$(\ell - 3v - \log_q(N))^2 > 4v^2 + 4v\log_q(N)$$

and consequently, after squaring $2v$ out of the left binomial,

$$(\ell - v - \log_q(N))^2 > 4v(\ell - v).$$

We therefore get,

$$\ell - \log_q(N) > 2\sqrt{v(\ell - v)} + v = \ell \cdot \left(2\sqrt{\frac{v}{\ell} \cdot \left(1 - \frac{v}{\ell}\right)} + \frac{v}{\ell}\right).$$

Now, from Lemma 23 the following inequality holds:

$$\mathcal{H}_q\left(\frac{v}{\ell}\right) \leqslant 2\log_q(2)\sqrt{\frac{v}{\ell}\left(1 - \frac{v}{\ell}\right)} + \frac{v}{\ell},$$

and hence the following can be deduced

$$\ell > \log_q(N) + \ell \cdot \mathcal{H}_q\left(\frac{\tau_{\mathsf{d}} - 1}{\ell}\right) = \log_q\left(N \cdot q^{\ell \cdot \mathcal{H}_q\left(\frac{\tau_{\mathsf{d}}-1}{\ell}\right)}\right).$$

Lastly

$$q^{\ell} > N \cdot B_{\tau_{\mathsf{d}}-1}(\ell),$$

using the inequality $B_{\tau_{\mathsf{d}}-1}(\ell) \leqslant q^{\ell \cdot \mathcal{H}_q\left(\frac{\tau_{\mathsf{d}}-1}{\ell}\right)}$. ∎

Corollary 25: Let $x = \log_q(M) + 3\tau_{\mathsf{i}}\log_q\log_q(M) + 3\tau_{\mathsf{i}} + 3$. For all

$$\begin{aligned}\tau_{\mathsf{d}} &\leqslant \frac{3}{10} \cdot L_M - \frac{1}{5} \cdot \left(\log_q(M) + 3\tau_{\mathsf{i}}\log_q\log_q(M) + 3\tau_{\mathsf{i}} + 3\right) \\ &= \frac{3 - 5\beta}{10} \cdot L - \frac{3}{5} \cdot \left(\log_q(\beta L) + \tau_{\mathsf{i}} + 1\right)\end{aligned}$$

and $L \geqslant \sqrt{2}x$, there exists an explicit construction of an $(\tau_{\mathsf{i}}, \tau_{\mathsf{d}})$-CCC using Algorithm 3 which uses a single symbol of redundancy.

Proof: Note that for

$$\begin{aligned}\tau_{\mathsf{d}} &\leqslant \frac{3}{10} \cdot L_M - \frac{1}{5} \cdot \left(\log_q(M) + 3\tau_{\mathsf{i}}\log_q\log_q(M) + 3\tau_{\mathsf{i}} + 3\right) \\ &= \frac{3}{10}L_M - \frac{1}{5}x \\ &= \frac{1}{80} \cdot (16L_M - 12x + 8L_M - 4x) \\ &= \frac{1}{80} \cdot \left(16L_M - 12x + 4\sqrt{(2L_M - x)^2}\right) \\ &\leqslant \frac{1}{80} \cdot \left(16L_M - 12x + 4\sqrt{5{L_M}^2 - 4L_M \cdot x - x^2}\right),\end{aligned}$$

it holds that

$$40{\tau_{\mathsf{d}}}^2 - (16L_M - 12x) \cdot \tau_{\mathsf{d}} + (L_M - x)^2 \leqslant 0.$$

In order to verify this inequality, we simply solve this quadratic equation and get

$$\tau_{\mathsf{d}1,2} = \frac{1}{80} \cdot \left(16L_M - 12x \pm \sqrt{\Delta}\right),$$

where

$$\begin{aligned}\Delta &= (16L_M - 12\ x)^2 - 160 \cdot (L_M - x)^2 \\ &= 96{L_M}^2 - 64L_M \cdot x - 16\ x^2 = 16(5{L_M}^2 - 4L_M \cdot x - x^2).\end{aligned}$$

Therefore $\tau_{\mathsf{d}} \leqslant \frac{1}{5}L_M - \frac{3}{20}x + \frac{1}{20} \cdot \sqrt{5{L_M}^2 - 4L_M \cdot x - x^2}$. Rearranging this inequality we get that

$$36{\tau_{\mathsf{d}}}^2 - 12\tau_{\mathsf{d}} \cdot (L_M - x) + (L_M - x)^2 \leqslant 4 \cdot \tau_{\mathsf{d}} \cdot (L_M - \tau_{\mathsf{d}}).$$

Observe that

$$36{\tau_{\mathsf{d}}}^2 - 12\tau_{\mathsf{d}} \cdot (L_M - x) + (L_M - x)^2 = (L_M - x - 6\tau_{\mathsf{d}})^2,$$

and that $L_M - x - 6\tau_{\mathsf{d}}$ is negative. Therefore

$$L_M - x - 5\tau_{\mathsf{d}} \geqslant 2\sqrt{\tau_{\mathsf{d}} \cdot (L_M - \tau_{\mathsf{d}})} + \tau_{\mathsf{d}}.$$

Recall that according to Lemma 23

$$\begin{aligned}&2\sqrt{\tau_{\mathsf{d}} \cdot (L_M - \tau_{\mathsf{d}})} + \tau_{\mathsf{d}} \\ &\geqslant L_M \cdot \left(2\log_q(2) \cdot \sqrt{\frac{\tau_{\mathsf{d}}}{L_M} \cdot \left(1 - \frac{\tau_{\mathsf{d}}}{L_M}\right)} + \frac{\tau_{\mathsf{d}}}{L_M}\right) \\ &\geqslant L_M \cdot \mathcal{H}_q\left(\frac{\tau_{\mathsf{d}}}{L_M}\right).\end{aligned}$$

Hence, $B_r(n) \leqslant q^{n \cdot \mathcal{H}_q\left(\frac{r}{n}\right)} \leqslant (n \cdot (q-1))^r$, and we get that

$$\begin{aligned}L_M - \log_q(M) &\geqslant 5\tau_{\mathsf{d}} + x - \log_q(M) + L_M \cdot \mathcal{H}_q(\frac{\tau_{\mathsf{d}}}{L_M}) \\ &\geqslant 5\tau_{\mathsf{d}} + 3\tau_{\mathsf{i}}\log_q\log_q(M) + 3\tau_{\mathsf{i}} + 3 + \log_q(B_2) \\ &\geqslant 5\tau_{\mathsf{d}} + 3\log_q(B_1) + \log_q(B_2) + 3.\end{aligned}$$

Lastly, from Theorem 19 the value of τ_{d} should satisfy

$$\ell + \log_q(B_{\tau_{\mathsf{i}}}(\log_q(M)) \cdot B_{\tau_{\mathsf{d}}-1}(L_M)) + 3 \leqslant L - 2\log_q(M),$$

which concludes the proof with $\ell = 5\tau_{\mathsf{d}} + 2\log_q(B_1)$ that is computed in Lemma 24. ∎

According to Section IV, $r_{M,L}(\tau_{\mathsf{i}}, \tau_{\mathsf{d}}) = 1$ when τ_{d} is approximately $L_M \cdot \mathcal{H}_q^{-1}\left(\frac{1-2\beta}{1-\beta}\right)$. However, this is not achieved by an explicit construction of such codes. Here, we presented an explicit construction in which the maximum value of τ_{d} is roughly $\frac{3L_M - 2\beta}{10}$.

The results from Corollary 15 and Corollary 25 are presented in Figures 8 and 9 for some fixed values of β, L, and τ_{i}.

Remark 5: Based on the results, and in particular Figure 9, in [31], the error rate of substitutions is approximately $1.32 \cdot 10^{-3}$. For $L = 160, \beta = \frac{1}{10}$, the expectation of the number of errors in the index field and data field of the strands is 0.02112 and 0.19008 respectively. We can use $\tau_i = 8$, and according to Corollary 25 achieve $\tau_d \leqslant 33.4$. According to Theorem 7, if the dominance property is satisfied, we can handle up to 8 errors in the data field and 3 errors in the index field. Hence, we can support any (t_i, t_d)-DNA system such that $t_i \leqslant 3$ and $t_d \leqslant 8$. In case the dominance property is not satisfied we can still support any (t_i, t_d)-DNA system such that $t_i \leqslant 1$ and $t_d \leqslant 8$, which is 40 times more than the expectation.

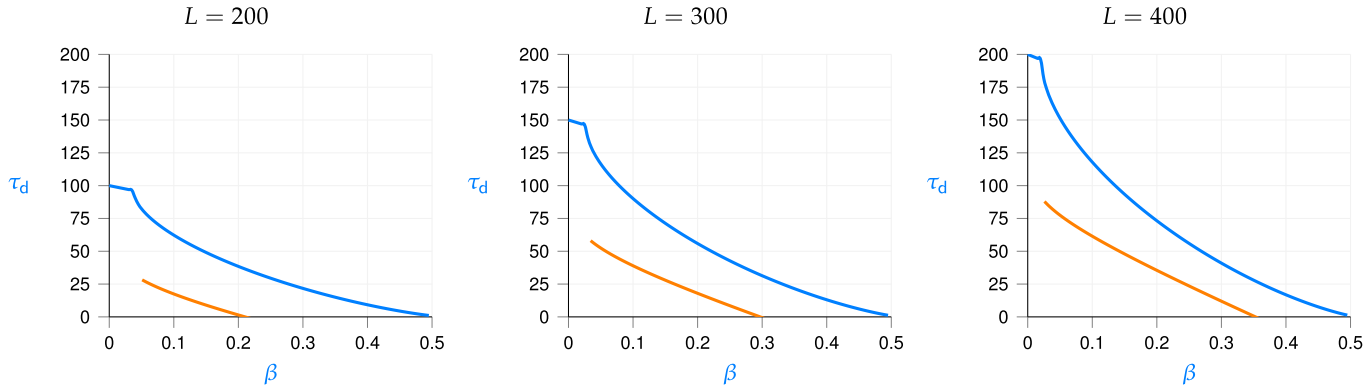

Fig. 8. Corollary 15 (in blue) and Corollary 25 (in orange) for $\tau_{\mathsf{i}} = 10$ over binary alphabet $p = 2$. The blue graph represents the upper bound on the value of τ_{d} when using a single symbol of redundancy, while the orange graph represents the value of τ_{d} can be achieved using the construction described in Algorithm 3.

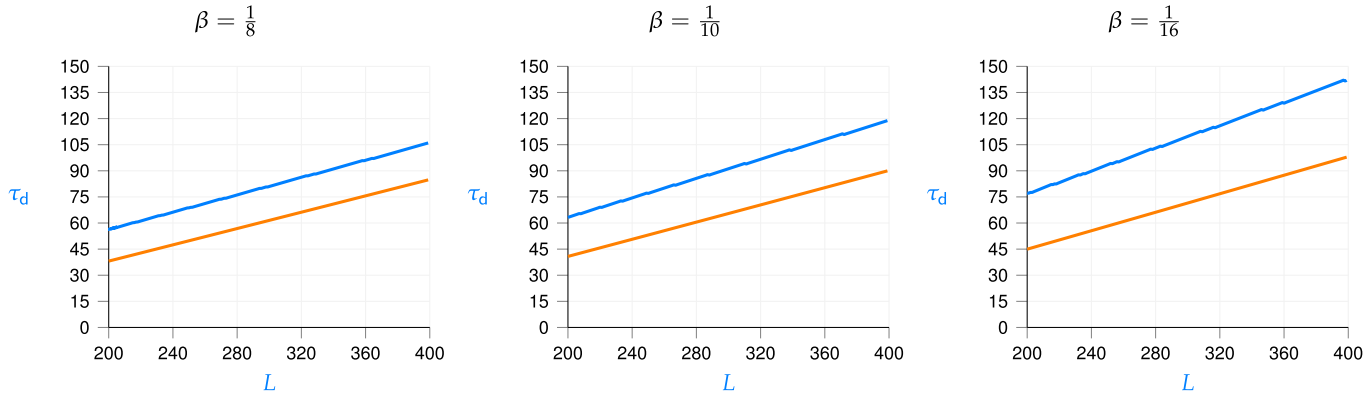

Fig. 9. Corollary 15 (in blue) and Corollary 25 (in orange) for $\tau_{\mathsf{i}} = 10$ over binary alphabet $p = 2$. The blue graph represents the upper bound on the value of τ_{d} when using a single symbol of redundancy, while the orange graph represents the value of τ_{d} can be achieved using the construction described in Algorithm 3.

VI. Edit Distance Extensions

In this section we show how the concept of CCCs can be extended to support errors in the form of insertions, deletions, and substitutions. So far we considered only substitution errors while data stored in DNA storage systems can also be read with insertions and deletions. Most of our results in the paper so far are independent of the type of errors that occur in the data. Yet, there are some results that should be revisited and extended for the edit distance.

First we define the edit distance between two vectors $\boldsymbol{x}$ and $\boldsymbol{y}$ to be the minimum number of insertion, deletion and substitution operations required to transform $\boldsymbol{x}$ into $\boldsymbol{y}$. We denote this distance by $\mathrm{d}_{\mathsf{E}}(\boldsymbol{x}, \boldsymbol{y})$. The radius-$r$ edit ball of a vector $\boldsymbol{x} \in [q]^n$ is $B_r^E(\boldsymbol{x}) = \{\boldsymbol{y} \mid \mathrm{d}_{\mathsf{E}}(\boldsymbol{x}, \boldsymbol{y}) \leqslant r\}$ and note that the size of $B_r^E(\boldsymbol{x})$ depends on the vector $\boldsymbol{x}$. We denote the size of the largest radius-r edit ball over all vectors of length ℓ by $B_r^E(\ell)$. For the rest of this section, the radius-r ball $B_r(\boldsymbol{x})$ for substitutions of a length-n vector $\boldsymbol{x}$ defined in Section II will be referred as $B_r^H(\boldsymbol{x})$ and its size will be referred as $B_r^H(n)$.

Next, recall that we have defined a $(t_{\mathsf{i}}, t_{\mathsf{d}})$-DNA system in Definition 1. The meaning of t_{i} and t_{d} has to change slightly when shifting to the edit distance as follows.

Definition 26: A DNA-based storage system is called a $(t_{\mathsf{i}}, t_{\mathsf{d}})^E$-DNA system if it satisfies the property that for each $\boldsymbol{v}' = (\mathsf{ind}', \boldsymbol{u}')$ which is a noisy copy of the input strand $\boldsymbol{v} = (\mathsf{ind}, \boldsymbol{u})$, it holds that

$$\mathrm{d}_{\mathsf{E}}\left(v_{[0,\log_q(M)]}, v'_{[0,\log_q(M)]}\right) \leqslant t_{\mathsf{i}}$$

and

$$\mathrm{d}_{\mathsf{E}}\left(v_{[\log_q(M), L_M]}, v'_{[\log_q(M), |v'| - \log_q(M)]}\right) \leqslant t_{\mathsf{d}}.$$

Lastly, the definition of CCCs from Definition 4 must also be modified to support the edit distance. The definition itself is almost identical, and a proper version for the edit distance is provided by replacing all occurrences of d_{H} with d_{E}. The new codes will be referred as $(\tau_{\mathsf{i}}, \tau_{\mathsf{d}})_E$-CCCs. As a result of this definition, we define also $A_{M,L}^E(\tau_{\mathsf{i}}, \tau_{\mathsf{d}})$ to be the size of the largest $(\tau_{\mathsf{i}}, \tau_{\mathsf{d}})_E$-CCC and $r_{M,L}^E(\tau_{\mathsf{i}}, \tau_{\mathsf{d}})$ to be the optimal redundancy of the code. Throughout this section, we say that a theorem, lemma, or corollary is also valid for the edit distance if by replacing all occurrences of d_{H}, $B_r^H(n), A_{M,L}(\tau_{\mathsf{i}}, \tau_{\mathsf{d}}), r_{M,L}(\tau_{\mathsf{i}}, \tau_{\mathsf{d}})$ with d_{E}, $B_r^E(n), A_{M,L}^E(\tau_{\mathsf{i}}, \tau_{\mathsf{d}}), r_{M,L}^E(\tau_{\mathsf{i}}, \tau_{\mathsf{d}})$, the definitions, claims and proofs remain valid, respectively.

In Section III we have defined CCCs and the clustering constraint. We also proved the capabilities of CCCs in Theorem 7.

As the only property of the distance function used in this section is the triangular inequality, which applies also for the edit distance, the proof of Theorem 7 applies together with Algorithms 1 and 2 that are derived from it. Note that this is true because of the way we altered the definition of a DNA system such that comparing the index sized prefixes of the strands still obey the constraint of at most t_{i} substitutions.

Next, in Section IV we have derived lower and upper bounds on $A_{M,L}(\tau_{\mathsf{i}},\tau_{\mathsf{d}})$, which is the size of the largest CCC, in Theorems 12, 14 respectively. Also an asymptotic improvement of Theorem 14 for $\tau_{\mathsf{i}}=1$ was presented in Theorem 17. The three theorems do not rely on any property of the Hamming metric and therefore are true for the edit metric as well. In addition, for all $x\in[q]^n$ and r, it holds that $B_r^H(n)\leqslant B_r^E(n)$, and therefore Corollary 15, that relies on a lower bound of $B_r^H(n)$, is also valid for the edit distance. Corollary 13, on the other hand, has to be revisited and will be addressed in the next corollary.

Throughout the rest of the section the following upper bound is used. Its proof can be found in Appendix A

Lemma 27: For any r and $\boldsymbol{x}\in[q]^n$ it holds that

$$B_r^E(\boldsymbol{x})\leqslant 6^r\cdot B_r^H(n+r).$$

Next, we adopt Corollary 13 to the edit distance using the upper bound from Lemma 27. For the next corollary, B_1,B_2 get its own version for the edit distance denoted by $B_1^E=B_{\tau_{\mathsf{i}}}^E(\log_q(M))-1, B_2^E=B_{\tau_{\mathsf{d}}-1}^E(L_M)$, respectively.

Corollary 28: Denote by $x\triangleq(1+\mathsf{le}_q\cdot(M-E))\cdot B_1^E$. For $M=q^{\beta L}$ and all τ_{d} such that

$$\tau_{\mathsf{d}}\leqslant\frac{5}{8}L-\beta L-\frac{3}{8}\log_q(\mathsf{le}_q)-\frac{3}{8}\tau_{\mathsf{i}}\log_q(\beta L)-\frac{15}{8}\tau_{\mathsf{i}}$$

and $L\geqslant\log_q(x)$, it holds that $r_{M,L}^E(\tau_{\mathsf{i}},\tau_{\mathsf{d}})<1$.

Proof: Let $\tau_{\mathsf{d}}\leqslant\frac{5}{8}L-\beta L-\frac{3}{8}\left(\log_q(\mathsf{le}_q)-\tau_{\mathsf{i}}\log_q(\beta L)-5\tau_{\mathsf{i}}\right)$. It holds that

$$\begin{aligned}\tau_{\mathsf{d}}&\leqslant\frac{5}{8}L-\beta L-\frac{3}{8}\log_q(\mathsf{le}_q)-\frac{3}{8}\tau_{\mathsf{i}}\log_q(\beta L)-\frac{15}{8}\tau_{\mathsf{i}}\\&\leqslant\frac{5}{8}L_M-\frac{3}{8}\log_q(M)-\frac{3}{8}\log_q(\mathsf{le}_q)-\frac{3}{8}\tau_{\mathsf{i}}\log_q\left(2\beta L(6q-6)\right)\\&\leqslant\frac{5}{8}L_M-\frac{3}{8}\log_q(M\cdot\mathsf{le}_q)-\frac{3}{8}\log_q\left(((\log_q(M)+\tau_{\mathsf{i}})(6q-6))^{\tau_{\mathsf{i}}}\right).\end{aligned}$$

From Lemma 27 we have

$$B_1^E\leqslant 6^e\cdot B_{\tau_{\mathsf{i}}}^H(\log_q(M)+\tau_{\mathsf{i}})\leqslant\left((6q-6)\cdot(\log_q(M)+\tau_{\mathsf{i}})\right)^{\tau_{\mathsf{i}}},$$

and hence

$$\begin{aligned}\tau_{\mathsf{d}}&\leqslant\frac{5}{8}L_M-\frac{3}{8}\log_q\left((1+\mathsf{le}_q\cdot(M-E))\cdot B_1^E\right)\\&=\frac{5}{8}L_M-\frac{3}{8}\log_q(x)\\&=\frac{3}{8}L_M-\frac{1}{4}\log_q(x)+\frac{1}{8}\cdot\sqrt{\left(2L_M-\log_q(x)\right)^2}\\&\leqslant\frac{3}{8}L_M-\frac{1}{4}\log_q(x)+\frac{1}{8}\cdot\sqrt{5{L_M}^2-4L_M\cdot\log_q(x)}.\end{aligned}$$

From Theorem 12 it holds that

$$r_{M,L}^E(\tau_{\mathsf{i}},\tau_{\mathsf{d}})<\frac{\mathsf{le}_q\cdot(M-E)B_1^EB_2^E}{q^{L_M}-B_1^EB_2^E}.$$

Thus, it is enough to prove that t satisfies $\frac{\mathsf{le}_q\cdot(M-E)B_1^EB_2^E}{q^{L_M}-B_1^EB_2^E}<1$.

For $v=\tau_{\mathsf{d}}-1$ we seek to show the following inequality

$$16v^2-\left(12L_M-8\log_q(x)\right)\cdot v+\left(L_M-\log_q(x)\right)^2\leqslant 0.$$

In order to verify this inequality, we simply solve this quadratic equation. Its two solutions are

$$v_{1,2}=\frac{1}{32}\cdot\left(12L_M-8\log_q(x)\pm\sqrt{\Delta}\right),$$

where

$$\begin{aligned}\Delta&=\left(12L_M-8\log_q(x)\right)^2-64\cdot\left(L_M-\log_q(x)\right)^2\\&=144{L_M}^2-192L_M\cdot\log_q(x)-64{L_M}^2+128L_M\cdot\log_q(x)\\&=80{L_M}^2-64L_M\cdot\log_q(x)=16\left(5{L_M}^2-4L_M\cdot\log_q(x)\right).\end{aligned}$$

Therefore $v\leqslant\frac{3}{8}L_M-\frac{1}{4}\log_q(x)+\frac{1}{8}\cdot\sqrt{5{L_M}^2-4L_M\cdot\log_q(x)}$. Rearranging this inequality we get that

$$16\,v^2-8v\cdot\left(L_M-\log_q(x)\right)+\left(L_M-\log_q(x)\right)^2\leqslant 4\cdot v\cdot L_M.$$

Observe that the left hand side is actually $\left(L_M-\log_q(x)-4v\right)^2$, and that $L_M-\log_q(x)-4v$ is negative. Therefore

$$L_M-\log_q(x)-3v\geqslant 2\sqrt{v\cdot L_M}+v.$$

Recall that according to Lemma 23

$$\begin{aligned}&2\sqrt{v\cdot L_M}+v\\&\geqslant(L_M+v)\cdot\left(2\cdot\sqrt{\frac{v}{L_M+v}\cdot\left(1-\frac{v}{L_M+v}\right)}+\frac{v}{L_M+v}\right)\\&\geqslant(L_M+v)\cdot\mathcal{H}_q\left(\frac{v}{L_M+v}\right).\end{aligned}$$

Hence, using

$$B_r^E(n)\leqslant 6^r\cdot B_r^H(n+r)\leqslant 6^r\cdot q^{(n+r)\cdot\mathcal{H}_q\left(\frac{r}{n+r}\right)},$$

we get

$$\begin{aligned}L_M-\log_q(x)&\geqslant 2\sqrt{v\cdot L_M}+v+3v\\&>2\sqrt{v\cdot L_M}+v+\log_q(6)\cdot v\\&>\log_q\left(B_v^E(L_M)\right)=\log_q(B_2^E),\end{aligned}$$

and finally

$$B_2^E\cdot x=(1+\mathsf{le}_q\cdot(M-E))\cdot B_1^E\cdot B_2^E<q^{L_M}.$$

After rearranging we get, $\frac{\mathsf{le}_q\cdot(M-E)B_1^EB_2^E}{q^{L_M}-B_1^EB_2^E}<1$ as required. ∎

In Section V, a construction of CCCs using a single symbol of redundancy has been presented. The construction itself is valid and applies also for the edit distance but requires redefining properly the functions $\Delta_1,\Delta_2,w_\ell(S,t)$.

The function definitions are updated as follows.

- The function $w_\ell(S,t)$ is defined over a set of vectors S and a positive integer t and outputs a vector $\boldsymbol{w}\in[q]^\ell$ which satisfies the following condition. For all $\boldsymbol{v}\in S$, $\mathrm{d}_{\mathsf{E}}(\boldsymbol{w},\boldsymbol{v}_{[\log_q(M),\ell]})\geqslant t$. The value of ℓ for the edit

Algorithm 4 Construction of $\boldsymbol{w}_\ell(S,t)$ for Edit Distance

Input: A set $S=\{\boldsymbol{v}_0,\ldots,\boldsymbol{v}_k\}, \boldsymbol{v}_i\in[q]^{2t\cdot\left(\log_q(|S|)+1\right)}$
Output: A vector $\boldsymbol{w}\leftarrow w_\ell(S,t)$
1: $\boldsymbol{w}\leftarrow 0^{2t\cdot\left(\log_q(|S|)+1\right)}$, $\mathsf{cursor}\leftarrow\log_q(M)$
2: **while** $S\neq\emptyset$ **do**
3: $\quad Q\leftarrow\left\{\left(c,\{\boldsymbol{v}_i|\boldsymbol{v}_i\in S, w_{\#c}((\boldsymbol{v}_i)_{[\mathsf{cursor},2t]})>t\}\right)|c\in[q]\right\}$
4: $\quad c_{\min},S\leftarrow\underset{(c,S_c)\in Q}{\operatorname{argmin}}|S_c|$
5: $\quad \boldsymbol{w}_{[\mathsf{cursor},2t]}\leftarrow(c_{\min})^{2t}$
6: $\quad \mathsf{cursor}\leftarrow\mathsf{cursor}+2t$
7: **end while**

distance version of this function will be determined later as a function of τ_{i}, τ_{d}, and M.

- The function $\Delta_1(\mathsf{ind}_i,\mathsf{ind}_j)$ encodes the difference between the two indices i and j of edit distance at most τ_{i} using $\lceil\log(B^E_{\tau_{\mathsf{i}}}(\log_q(M)))\rceil$ symbols which encode the index of ind_j in a lexicographic enumeration of $B^E_{\tau_{\mathsf{i}}}(\mathsf{ind}_i)$.
- The function $\Delta_2(\boldsymbol{u}_i,\boldsymbol{u}_j)$ encodes the difference between the two data fields $\boldsymbol{u}_i,\boldsymbol{u}_j\in[q]^{L_M}$ of edit distance at most $\tau_{\mathsf{d}}-1$ using $\lceil\log(B^E_{\tau_{\mathsf{d}}-1}(L_M))\rceil$ symbols which encode the index of $\boldsymbol{u}_j$ in a lexicographic enumeration of $B^E_{\tau_{\mathsf{d}}-1}(\boldsymbol{u}_i)$.

The edit distance version of $w_\ell(s,t)$ requires a new construction and an analysis as Lemma 21 and Lemma 24 cannot be used under the edit distance. Before getting into computing and analyzing $w_\ell(S,t)$, the fact that embedding $w_\ell(S,t)$ into a vector preserves the distance property has to be justified for the correctness of the construction. This was trivial when Theorem 19 was proved for substitutions only. Hence, we need to prove the following property. The proof of this property is deferred to Appendix A.

Lemma 29: Let $\boldsymbol{x},\boldsymbol{y}\in[q]^n$ and let $m,i,t\in[n]$ be such that $\mathrm{d}_{\mathsf{E}}(\boldsymbol{x}_{[i,m]},\boldsymbol{y}_{[i,m]})\geqslant t$. Then, it also holds that $\mathrm{d}_{\mathsf{E}}(\boldsymbol{x},\boldsymbol{y})\geqslant t$.

Now, we provide a new efficient construction for the vector $w_\ell(S,t)$ for $\ell=2t\cdot\left(\log_q(|S|)+1\right)$, which is roughly twice from the result for w_ℓ in its version for the Hamming distance. As before, this construction is not optimal in the output's length. However, it does improve the encoding complexity of Algorithm 3 with respect to the one presented in Theorem 20. The proposed algorithm is presented in Algorithm 4.

The next claim will be used in the theorem which verifies the correctness of Algorithm 4.

Claim 1: Let $\boldsymbol{x}\in[q]^n$ and $s\in[q]$ such that $w_{\#s}(\boldsymbol{x})\leqslant n/2$. Then, it holds that $\mathrm{d}_{\mathsf{E}}(\boldsymbol{x},s^n)\geqslant n/2$.

Proof: Assume on the contrary that $\mathrm{d}_{\mathsf{E}}(\boldsymbol{x},s^n)<n/2$. That is, a sequence of less than $n/2$ editions can be found to transform $\boldsymbol{x}$ to a vector of the same length containing only the symbol $s\in[q]$. On the other hand, $w_{\#s}(\boldsymbol{x})\leqslant n/2$, and therefore to reach n occurrences of the symbol s we have to introduce at least $n/2$ new appearances. This is not possible as each operation increases the amount of occurrences of the symbol s by at most 1 and there are less than $n/2$ operations in the sequence. This contradiction concludes the proof. ∎

We are now ready to prove the correctness of Algorithm 4 and its complexity.

Theorem 30: The output of Algorithm 4 is a valid computation of $w_\ell(S,t)$ for $\ell=2t\cdot\left(\log_q(|S|)+1\right)$. The complexity of the construction process is $O(t\cdot|S|\cdot\log_q(|S|))$ symbol operations.

Proof: Denote $w_\ell(S,t)$ by $\boldsymbol{w}$. The goal is to find efficiently a vector $\boldsymbol{w}$ such that for all $\boldsymbol{v}\in S_{[\log_q(M),\ell]}$ it holds that $\mathrm{d}_{\mathsf{E}}(\boldsymbol{w},v)\geqslant t$. In order to achieve this, the vector $\boldsymbol{w}$ is built iteratively, with sequences of length $2t$ which consists of a single symbol.

In Step 3 the algorithm computes for each $c\in[q]$ the vectors in S that have more than t appearances of c. Let $\boldsymbol{v}_i\in S$. Observe that if $w_{\#c}\left((\boldsymbol{v}_i)_{[\mathsf{cursor},2t]}\right)>t$, then for any $c\neq c'\in[q]$ it holds that $w_{\#c}\left((\boldsymbol{v}_i)_{[\mathsf{cursor},2t]}\right)<t$, and thus it is concluded that $S_{c'}\cap S_c=\emptyset$.

Also, if $c=c_{\min}$ by Claim 1 it holds that

$$\mathrm{d}_{\mathsf{E}}((\boldsymbol{v}_i)_{[\mathsf{cursor},2t]},c^{2t})\geqslant t.$$

Hence, $\mathrm{d}_{\mathsf{E}}((\boldsymbol{v}_i)_{[\mathsf{cursor},2t]},\boldsymbol{w}_{[\mathsf{cursor},2t]})\geqslant t$ and by Lemma 29 it is also true that $\mathrm{d}_{\mathsf{E}}(\boldsymbol{w},(\boldsymbol{v}_i)_{[\log_q(M),\ell]})\geqslant t$. Hence, in Step 4 the set S is updated in a way that any vector $\boldsymbol{v}_i$ that was not handled yet remains in S. That is, if the loop terminates, the requirement $\mathrm{d}_{\mathsf{E}}(\boldsymbol{w},(\boldsymbol{v}_i)_{[\log_q(M),\ell]})\geqslant t$ holds for all $\boldsymbol{v}_i\in S$.

For the complexity, each iteration of the while loop in Step 2 consists of a single heavy operation. This is done to compute the most dominant symbol in each $v_i\in S$. This can be done in linear time, and hence $O(2t\cdot|S|)$ symbol operations.

In Step 4 the algorithm picks the symbol to use for the next sequence. This operation requires to compute a minimum over a set and can be done with $O(\log_2(q))$ symbol operations. The symbol is picked in a way that the number of members $v_i\in S$ that remain in S is minimized, and therefore the size of S decreases by a factor of $1/q$. Otherwise for all S_c it holds that $|S_c|>\frac{|S|}{q}$ and thus

$$|S|\geqslant\left|\bigcup_{c\in[q]}S_c\right|=\sum_{c\in[q]}|S_c|>q\cdot\frac{|S|}{q}>|S|,$$

and that would be a contradiction. It also provides a justification for $\ell=2t\cdot\left(\log_q(|S|)+1\right)$. As the size of S decreases by $1/q$ each iteration, after $\log_q(|S|)+1$ the while loop in Step 2 will terminate. We get an overall of

$$(2t\cdot|S|)\cdot(\log_q(|S|)+1)=O(t\cdot|S|\cdot\log_q(|S|))$$

symbol operations. ∎

The following lemma suggests an upper bound on the optimal size of $w_\ell(S,t)$ for the edit distance. Similarly to the Hamming distance case, this optimal length will be denoted by $\ell^E(e,t,M)$.

Lemma 31: For all $\tau_{\mathsf{i}},\tau_{\mathsf{d}},M$ it holds that

$$\ell^E(\tau_{\mathsf{i}},\tau_{\mathsf{d}},M)\leqslant 12\tau_{\mathsf{d}}+2\tau_{\mathsf{i}}\cdot\log_q(\log_q(M)\ +\tau_{\mathsf{i}})+2\tau_{\mathsf{i}}\cdot\log_q(6q-6).$$

Proof: We start by denoting the size of $S_{[\log_q(M),\ell]}$ by N. We show that for $\ell\geqslant 12\tau_{\mathsf{d}}+2\log_q(N)-12$ the following inequality is satisfied

$$q^\ell>N\cdot B^E_{\tau_{\mathsf{d}}-1}(\ell),$$

and therefore for $N = B^E_{\tau_\mathsf{i}}(\log_q(M)) - 1$, using Lemma 27 we have

$$\begin{aligned} B^E_{\tau_\mathsf{i}}(\log_q(M)) &\leqslant 6^{\tau_\mathsf{i}} \cdot B^H_{\log_q(M)+\tau_\mathsf{i}}(\tau_\mathsf{i}) \\ &\leqslant 6^{\tau_\mathsf{i}} \left((\log_q(M)+\tau_\mathsf{i})\cdot(q-1)\right)^{\tau_\mathsf{i}}, \end{aligned}$$

and hence

$$\begin{aligned} \log_q(N) &= \log_q(B^E_{\tau_\mathsf{i}}(\log_q(M))) \\ &\leqslant \tau_\mathsf{i}\cdot\log_q(6) + \tau_\mathsf{i}\cdot\log_q\left((\log_q(M)+\tau_\mathsf{i})\cdot(q-1)\right). \end{aligned}$$

This provides the desired result in which

$$\begin{aligned} \ell^E(\tau_\mathsf{i},\tau_\mathsf{d},M) &\leqslant 12\tau_\mathsf{d} + 2\log_q(N) - 12 \\ &\leqslant 12t + 2\tau_\mathsf{i}\cdot\log_q(\log_q(M)+\tau_\mathsf{i}) + 2\tau_\mathsf{i}\cdot\log_q(6q-6). \end{aligned}$$

Let $\ell \geqslant 12\tau_\mathsf{d} + 2\log_q(N) - 12$ and $v = \tau_\mathsf{d} - 1$.

Using the inequality of arithmetic and geometric means, $2\sqrt{a\cdot b} \leqslant a+b$ (where equality holds if and only if $a=b$) it holds that

$$\ell \geqslant 12v + 2\log_q(N) \geqslant 6v + \log_q(N) + 2\sqrt{v\cdot\left(5v+\log_q(N)\right)}$$

with $a = v$, $b = v + 5\log_q(N)$ and note that $a \neq b$. Rearranging this inequality and squaring both sides we get that

$$(\ell - 6v - \log_q(N))^2 > 20v^2 + 4v\log_q(N)$$

and consequently, after squaring $2v$ out of the left binomial,

$$(\ell - 4v - \log_q(N))^2 > 4v\ell.$$

we get,

$$\begin{aligned} &\ell - \log_q(N) > 2\sqrt{v\cdot\ell} + 4v \\ &= (\ell+v)\cdot\left(2\sqrt{\frac{v}{\ell+v}\cdot\left(1-\frac{v}{\ell+v}\right)} + \frac{v}{\ell+v}\right) + 3v. \end{aligned}$$

Now, from Lemma 23 the following inequality holds

$$\mathcal{H}_q\left(\frac{v}{\ell+v}\right) \leqslant 2\log_q(2)\sqrt{\frac{v}{\ell+v}\left(1-\frac{v}{\ell+v}\right)} + \frac{v}{\ell+v},$$

and hence the following can be deduced

$$\begin{aligned} &\ell > \log_q(N) + (\ell+\tau_\mathsf{d}-1)\cdot\mathcal{H}_q\left(\frac{\tau_\mathsf{d}-1}{\ell+\tau_\mathsf{d}-1}\right) + 3(\tau_\mathsf{d}-1) \\ &> \log_q\left(N\cdot 6^{\tau_\mathsf{d}-1}\cdot q^{(\ell+\tau_\mathsf{d}-1)\cdot\mathcal{H}_q\left(\frac{\tau_\mathsf{d}-1}{\ell+\tau_\mathsf{d}-1}\right)}\right). \end{aligned}$$

Lastly with $\log_q(6) < 3$

$$q^\ell > N\cdot B^E_{\tau_\mathsf{d}-1}(\ell),$$

and by using the inequality from Lemma 27 we get that

$$B^E_{\tau_\mathsf{d}-1}(\ell) \leqslant 6^{(\tau_\mathsf{d}-1)}\cdot B^H_{\ell+\tau_\mathsf{d}-1}(\tau_\mathsf{d}-1) \leqslant 6^{(\tau_\mathsf{d}-1)}\cdot q^{(\ell+\tau_\mathsf{d}-1)\cdot\mathcal{H}_q\left(\frac{\tau_\mathsf{d}-1}{\ell+\tau_\mathsf{d}-1}\right)}.$$

■

At last, we can derive an upper bound for the value of τ_d that can be achieved using our construction in the edit distance case.

Corollary 32: Let $x = \log_q(M) + 3\tau_\mathsf{i}\log_q\log_q(M) + 15\tau_\mathsf{i} + 3$. For all

$$\begin{aligned} \tau_\mathsf{d} &\leqslant \frac{5}{64}\cdot L_M - \frac{1}{16}\cdot\left(\log_q(M)+3\tau_\mathsf{i}\log_q\log_q(M)+15\tau_\mathsf{i}+3\right) \\ &= \frac{5-9\beta}{64}\cdot L - \frac{3}{16}\cdot\left(\log_q(\beta L)+5\tau_\mathsf{i}+1\right) \end{aligned}$$

and $L \geqslant x$, there exists an explicit construction of an $(\tau_\mathsf{i},\tau_\mathsf{d})$-CCC using Algorithm 3 which uses a single symbol of redundancy.

Proof: Note that for

$$\begin{aligned} \tau_\mathsf{d} &\leqslant \frac{5}{64}\cdot L_M - \frac{1}{16}\cdot\left(\log_q(M)+3\tau_\mathsf{i}\log_q\log_q(M)+15\tau_\mathsf{i}+3\right) \\ &= \frac{5}{64}L_M - \frac{1}{16}x \\ &= \frac{1}{512}\cdot(36L_M - 32x + 4L_M) \\ &= \frac{1}{512}\cdot\left(36L_M - 32x + 4\sqrt{{L_M}^2}\right) \\ &\leqslant \frac{1}{512}\cdot\left(36L_M - 32x + 4\sqrt{17{L_M}^2 - 16L_M\cdot x}\right), \end{aligned}$$

it holds that

$$256{\tau_\mathsf{d}}^2 - (36L_M - 32x)\cdot\tau_\mathsf{d} + (L_M - x)^2 \leqslant 0.$$

In order to verify this inequality, we simply solve this quadratic equation

$$\tau_{\mathsf{d}1,2} = \frac{1}{512}\cdot\left(36L_M - 32x \pm \sqrt{\Delta}\right),$$

where

$$\begin{aligned} \Delta &= (36L_M - 32\ x)^2 - 1024\cdot(L_M - x)^2 \\ &= 1296{L_M}^2 - 2304L_M\cdot x - 1024{L_M}^2 + 2048L_M\cdot x \\ &= 272{L_M}^2 - 256L_M\cdot x = 16(17{L_M}^2 - 16L_M\cdot x). \end{aligned}$$

Therefore $\tau_\mathsf{d} \leqslant \frac{9}{128}L_M - \frac{1}{16}x + \frac{1}{128}\cdot\sqrt{17{L_M}^2 - 16L_M\cdot x}$. Rearranging this inequality we get that

$$256{\tau_\mathsf{d}}^2 - 32\tau_\mathsf{d}\cdot(L_M - x) + (L_M - x)^2 \leqslant 4\cdot\tau_\mathsf{d}\cdot L_M.$$

Observe that

$$256{\tau_\mathsf{d}}^2 - 32\tau_\mathsf{d}\cdot(L_M - x) + (L_M - x)^2 = (L_M - x - 16\tau_\mathsf{d})^2,$$

and that $L_M - x - 16\tau_\mathsf{d}$ is negative. Therefore

$$L_M - x - 15\tau_\mathsf{d} \geqslant 2\sqrt{\tau_\mathsf{d}\cdot L_M} + \tau_\mathsf{d}.$$

Recall that according to Lemma 23

$$\begin{aligned} &2\sqrt{\tau_\mathsf{d}\cdot L_M} + \tau_\mathsf{d} \\ &\geqslant (L_M+\tau_\mathsf{d})\cdot\left(2\cdot\sqrt{\frac{\tau_\mathsf{d}}{L_M+\tau_\mathsf{d}}\cdot\left(1-\frac{\tau_\mathsf{d}}{L_M+\tau_\mathsf{d}}\right)} + \frac{\tau_\mathsf{d}}{L_M+\tau_\mathsf{d}}\right) \\ &\geqslant (L_M+\tau_\mathsf{d})\cdot\mathcal{H}_q\left(\frac{\tau_\mathsf{d}}{L_M+\tau_\mathsf{d}}\right). \end{aligned}$$

Hence, using

$$\begin{aligned} B^E_r(n) &\leqslant 6^r\cdot B^H_r(n+r) \leqslant 6^r\cdot q^{(n+r)\cdot\mathcal{H}_q\left(\frac{r}{n+r}\right)} \\ &\leqslant \left((n+r)\cdot(q-1)\right)^r, \end{aligned}$$

we get

$$\begin{aligned}
&L_M - \log_q(M) \geqslant 15\tau_{\mathsf{d}} + x - \log_q(M) + (L_M + \tau_{\mathsf{d}}) \cdot \mathcal{H}_q(\frac{\tau_{\mathsf{d}}}{L_M + \tau_{\mathsf{d}}}) \\
&\geqslant 15t + 3\tau_{\mathsf{i}} \log_q \log_q(M) + 15\tau_{\mathsf{i}} + 3 + \log_q(B^H_{\tau_{\mathsf{d}}-1}(L_M + \tau_{\mathsf{d}} - 1)) \\
&\geqslant 12\tau_{\mathsf{d}} + 3\tau_{\mathsf{i}} \cdot \log_q \left(\log_q(M) + \tau_{\mathsf{i}}\right) + 12\tau_{\mathsf{i}} + \log_q(B^E_{\tau_{\mathsf{d}}-1}(L_M)) + 3 \\
&\geqslant 12\tau_{\mathsf{d}} + 3\tau_{\mathsf{i}} \cdot \log_q \left((\log_q(M) + \tau_{\mathsf{i}})(6q-6)\right) + \log_q(B^E_{\tau_{\mathsf{d}}-1}(L_M)) + 3 \\
&\geqslant 12\tau_{\mathsf{d}} + 3\log_q(B^E_{\tau_{\mathsf{i}}}(\log_q(M))) + \log_q(B^E_{\tau_{\mathsf{d}}-1}(L_M)) + 3.
\end{aligned}$$

From Theorem 19 the value of t should satisfy

$$\ell + \log_q(B^E_{\tau_{\mathsf{i}}}(\log_q(M)) \cdot B^E_{\tau_{\mathsf{d}}-1}(L_M)) + 3 \leqslant L - 2\log_q(M).$$

Therefore using $\ell = 12\tau_{\mathsf{d}} + 2\log_q(B_1)$ computed in Lemma 31 concludes the proof. ∎

VII. Conclusion

In this paper we have presented a new family of codes, called clustering-correcting codes. These codes are beneficial in DNA-based storage systems in order to cluster the strands in the correct groups. We showed upper and lower bound on these codes as well as an explicit construction which uses a single symbol of redundancy. We then discussed extensions of those results from the Hamming distance metric into the edit distance metric.

Appendix A

Lemma 23.: Let $x \in [0,1]$ it holds that

$$4\log_q(2) \cdot x(1-x) \leqslant \mathcal{H}_q(x) \leqslant 2\log_q(2)\sqrt{x(1-x)} + x.$$

Proof: Let $g(x) = -x\log_q(x)$ and also let $f(x) = g(x) + g(1-x)$. Note that $f(x) = f(1-x)$. Also note that $f'(x) = \log_q(1-x) - \log_q(x)$. We have the following two derivatives:

1)

$$\begin{aligned}
\frac{d}{dx}\left(\frac{(f(x))^2}{x(1-x)}\right) &= \frac{2x(1-x)f(x)f'(x) - (1-2x)(f(x))^2}{x^2(1-x)^2} \\
&= f(x) \cdot \frac{-x\log_q(x) + (1-x)\log_q(1-x)}{x^2(1-x)^2} \\
&= \frac{(g(x))^2 - (g(1-x))^2}{x^2(1-x)^2}.
\end{aligned}$$

2)

$$\begin{aligned}
\frac{d}{dx}\left(\frac{f(x)}{4x(1-x)}\right) &= \frac{4x(1-x)f'(x) - 4(1-2x)f(x)}{16x^2(1-x)^2} \\
&= \frac{(1-x)^2\log_q(1-x) - x^2\log_q(x)}{4x^2(1-x)^2} \\
&= \frac{x \cdot g(x) - (1-x) \cdot g(1-x)}{x^2(1-x)^2}.
\end{aligned}$$

For all $x \in (0, \frac{1}{2})$ it holds that $g(x) > g(1-x)$. We can deduce that the first derivative is positive and therefore it has its maximum at $x = 1/2$. Hence,

$$\frac{(f(x))^2}{x(1-x)} < \frac{\left(f\left(\frac{1}{2}\right)\right)^2}{\frac{1}{2}(1-\frac{1}{2})} = \frac{\left(\log_q(2)\right)^2}{1/4}$$

and therefore $f(x) < 2\log_q(2)\sqrt{x(1-x)}$

Also, for all $x \in (0, \frac{1}{2})$ it holds that

$$x \cdot g(x) - (1-x) \cdot g(1-x) < 0.$$

That is, the second derivative is negative and reaches its minimum for $x = 1/2$. Therefore,

$$\frac{f(x)}{4x(1-x)} > \frac{f\left(\frac{1}{2}\right)}{4 \cdot \frac{1}{2}(1-\frac{1}{2})} = \log_q(2),$$

and hence $f(x) > 4\log_q(2) \cdot x(1-x)$.

All together we have $\mathcal{H}_q(x) = f(x) + x \cdot \log_q(q-1)$ so,

$$\begin{aligned}
\mathcal{H}_q(x) &= f(x) + x \cdot \log_q(q-1) < 2\log_q(2)\sqrt{x(1-x)} + x \\
\mathcal{H}_q(x) &= f(x) + x \cdot \log_q(q-1) > 4\log_q(2) \cdot x(1-x)
\end{aligned}$$

which confirms the lemma. ∎

Lemma 27.: For any r and $\boldsymbol{x} \in [q]^n$ it holds that

$$B^E_r(\boldsymbol{x}) \leqslant 6^r \cdot B^H_r(n+r).$$

Proof: The upper bound is based on the proof of Lemma 2.6 in [6]. According to this result it holds that for all $\boldsymbol{x} \in [q]^n$

$$B^E_r(\boldsymbol{x}) \leqslant (2q+2)^r \cdot \binom{n+r}{r}.$$

According to this bound, it is possible to bound the size of the edit distance ball with the Hamming distance ball as follows. For $q \geqslant 2$ it holds that

$$\begin{aligned}
B^E_r(\boldsymbol{x}) &\leqslant (2q+2)^r \cdot \binom{n+r}{r} \\
&= \left(\frac{2q+2}{q-1}\right)^r \cdot \binom{n+r}{r} \cdot (q-1)^r \\
&\leqslant \left(2 + \frac{4}{q-1}\right)^r \cdot B^H_r(n+r) \\
&\leqslant 6^r \cdot B^H_r(n+r),
\end{aligned}$$

which concludes the proof. ∎

Lemma 29.: Let $\boldsymbol{x}, \boldsymbol{y} \in [q]^n$ and let $m, i, t \in [n]$ be such that $\mathrm{d}_{\mathsf{E}}(\boldsymbol{x}_{[i,m]}, \boldsymbol{y}_{[i,m]}) \geqslant t$. Then, it also holds that $\mathrm{d}_{\mathsf{E}}(\boldsymbol{x}, \boldsymbol{y}) \geqslant t$.

Proof: Assume on the contrary that $\mathrm{d}_{\mathsf{E}}(\boldsymbol{x}, \boldsymbol{y}) < t$ then there is a sequence of edit operations $O = o_0, \ldots, o_k$ that alters $\boldsymbol{x}$ into $\boldsymbol{y}$, such that $k < t$. It is possible to create another sequence $O_{[i,m]}$ that transforms $\boldsymbol{x}_{[i,m]}$ into $\boldsymbol{y}_{[i,m]}$ which also consists of less than t operations and that would be a contradiction.

The sequence O is divided into 9 types of operations: e_1 insertions, d_1 deletions, and s_1 substitutions performed on $\boldsymbol{x}_{[0,i]}$. e_2 insertions, d_2 deletions, and s_2 substitutions performed on $\boldsymbol{x}_{[i,m]}$. Lastly, e_3 insertions, d_3 deletions, and s_3 substitutions performed on $\boldsymbol{x}_{[i+m,n-i-m]}$. The sequence $O_{[i,m]}$ of operations is obtained as follows.

First, the e_2 insertions, d_2 deletions and s_2 substitutions must be found in $O_{[i,m]}$. Next, if $e_1 - d_1 > 0$ then there are symbols that were shifted from $\boldsymbol{x}_{[0,i]}$ to $\boldsymbol{x}_{[i,m]}$, and hence we should add to $O_{[i,m]}$ a total of $e_1 - d_1$ insertions. These insertions should be the same symbols that shift into $\boldsymbol{x}_{[i,m]}$ when performing O. Otherwise $e_1 - d_1 \leqslant 0$. In this case some symbols (or none) are shifted in the opposite direction

and therefore we add $d_1 - e_1$ deletions into $O_{[i,m]}$. Symbols can shift also from $\boldsymbol{x}_{[i+m,n-i-m]}$ to $\boldsymbol{x}_{[i,m]}$ or in the opposite direction and therefore $|e_1 + e_2 - d_1 - d_2|$ more insertions or deletions are needed.

$O_{[i,m]}$ performs the same operations on $\boldsymbol{x}_{[i,m]}$ that are done by O and therefore at the end of the process we get $\boldsymbol{y}_{[i,m]}$. The sequence contains a total of

$$|e_1 - d_1| + s_2 + e_2 + d_2 |e_1 + e_2 - d_1 - d_2|$$

operations. Observing that $|\boldsymbol{x}| = |\boldsymbol{y}|$ is easy to deduce that the amount of insertions and deletions in O must be the same. Hence, $e_1 + e_2 + e_3 = d_1 + d_2 + d_3$, and therefore the total amount of operations is $|e_1 - d_1| + s_2 + e_2 + d_2 + |e_3 - d_3|$ which is, of course, less than t. and this concluded the proof. ∎

REFERENCES

[1] M. Blawat *et al.*, "Forward error correction for DNA data storage," *Proc. Comput. Sci.*, vol. 80, pp. 1011–1022, Jan. 2016.

[2] J. Bornholt, R. Lopez, D. M. Carmean, L. Ceze, G. Seelig, and K. Strauss, "A DNA-based archival storage system," in *Proc. 21st Int. Conf. Architectural Program. Lang. Operating Syst. (ASPLOS)*, Atlanta, GA, USA, Apr. 2016, pp. 637–649.

[3] Y. M. Chee, H. M. Kiah, and H. Wei, "Efficient and explicit balanced primer codes," 2019, *arXiv:1901.01023*.

[4] M. Cheraghchi, J. Ribeiro, R. Gabrys, and O. Milenkovic, "Coded trace reconstruction," in *Proc. IEEE Inf. Theory Workshop (ITW)*, Visby, Sweden, Aug. 2019, pp. 1–5.

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

[6] V. Climenhaga, D. Thompson, and K. Yamamoto, "Large deviations for system with non-uniform structure," *Trans. Amer. Math. Soc.*, vol. 369, no. 6, pp. 4167–4192, 2017.

[7] T. Cover, "Enumerative source encoding," *IEEE Trans. Inf. Theory*, vol. IT-19, no. 1, pp. 73–77, Jan. 1973.

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

[9] R. P. Feynman, "There's plenty of room at the bottom," *Eng. Sci.*, vol. 23, no. 5, pp. 22–36, Feb. 1960.

[10] 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.

[11] N. Goldman *et al.*, "Towards practical, high-capacity, low-maintenance information storage in synthesized DNA," *Nature*, vol. 494, pp. 77–80, Jan. 2013.

[12] R. Heckel, G. Mikutis, and R. N. Grass, "A characterization of the DNA data storage channel," 2018, *arXiv:1803.03322*.

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

[14] H. M. Kiah, G. J. Puleo, and O. Milenkovic, "Codes for DNA sequence profiles," *IEEE Trans. Inf. Theory*, vol. 62, no. 6, pp. 3125–3146, Jun. 2016.

[15] A. Lenz, Y. Liu, C. Rashtchian, P. H. Siegel, A. Wachter-Zeh, and E. Yaakobi, "Coding for efficient DNA synthesis," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Los Angeles, CA, USA, Jun. 2020, pp. 2903–2908.

[16] A. Lenz, P. H. Siegel, A. Wachter-Zeh, and E. Yaakobi, "Coding over sets for DNA storage," *IEEE Trans. Inf. Theory*, vol. 66, no. 4, pp. 2331–2351, Apr. 2020.

[17] M. Levy and E. Yaakobi, "Mutually uncorrelated codes for DNA storage," *IEEE Trans. Inf. Theory*, vol. 66, no. 6, pp. 3671–3691, Jun. 2019.

[18] L. Organick *et al.*, "Scaling up DNA data storage and random access retrieval," *Nature Biotechnol.*, vol. 36, pp. 242–248, Mar. 2018.

[19] C. Rashtchian *et al.*, "Clustering billions of reads for DNA data storage," in *Proc. NIPS*, 2017, pp. 3360–3371.

[20] R. M. Roth, *Introduction to Coding Theory*. Cambridge, U.K.: Cambridge Univ. Press, 2005.

[21] M. Schwartz and T. Etzion, "Two-dimensional cluster-correcting codes," *IEEE Trans. Inf. Theory*, vol. 51, no. 6, pp. 2121–2132, Jun. 2005.

[22] T. Shinkar, E. Yaakobi, A. Lenz, and A. Wachter-Zeh, "Clustering-correcting codes," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Paris, France, Jul. 2019, pp. 81–85.

[23] J. Sima, N. Raviv, and J. Bruck, "On coding over sliced information," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Paris, France, Jul. 2019, pp. 767–771.

[24] W. Song, K. Cai, and K. A. S. Immink, "Sequence-subset distance and coding for error control for DNA-based data storage," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Paris, France, Jul. 2019, pp. 86–90.

[25] R. Talyansky, T. Etzion, and R. M. Roth, "Efficient code constructions for certain two-dimensional constraints," *IEEE Trans. Inf. Theory*, vol. 45, no. 2, pp. 794–799, Mar. 1999.

[26] S. M. H. T. Yazdi *et al.*, "DNA-based storage: Trends and methods," *IEEE Trans. Mol. Biol. Multi-Scale Commun.*, vol. 1, no. 3, pp. 230–248, Sep. 2015.

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

[28] S. M. H. Tabatabaei Yazdi, Y. Yuan, J. Ma, H. Zhao, and O. Milenkovic, "A rewritable, random-access DNA-based storage system," *Sci. Rep.*, vol. 5, no. 1, p. 14138, Nov. 2015.

[29] R. N. Grass, R. Heckel, M. Puddu, D. Paunescu, and W. J. Stark, "Robust chemical preservation of digital information on DNA in silica with error-correcting codes," *Angew. Chem. Int. Ed.*, vol. 54, no. 8, pp. 2552–2555, Feb. 2015.

[30] A. Lenz, P. H. Siegel, A. Wachter-Zeh, and E. Yaakobi, "Anchor-based correction of substitutions in indexed sets," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Paris, France, Jul. 2019, pp. 757–761.

[31] O. Sabary, Y. Orlev, R. Shafir, L. Anavy, E. Yaakobi, and Z. Yakhini, "SOLQC: Synthetic oligo library quality control tool," *Bioinformatics*, vol. 37, no. 5, pp. 720–722, May 2021.

Tal Shinkar (Student Member, IEEE) received the B.Sc. degree in computer science from the Technion—Israel Institute of Technology, Haifa, Israel, in 2019. His research interests include information and coding theory with applications to DNA storage.

Eitan Yaakobi (Senior Member, IEEE) received the B.A. degrees in computer science and mathematics and the M.Sc. degree in computer science from the Technion—Israel Institute of Technology, Haifa, Israel, in 2005 and 2007, respectively, and the Ph.D. degree in electrical engineering from the University of California, San Diego, in 2011. From 2011 to 2013, he was a Post-Doctoral Researcher with the Department of Electrical Engineering, California Institute of Technology, and the Center for Memory and Recording Research, University of California, San Diego. He is currently an Associate Professor with the Computer Science Department, Technion—Israel Institute of Technology. His research interests include information and coding theory with applications to non-volatile memories, associative memories, DNA storage, data storage and retrieval, and private information retrieval. He received the Marconi Society Young Scholar in 2009 and the Intel Ph.D. Fellowship from 2010 to 2011.

Andreas Lenz (Member, IEEE) received the B.Sc. degree (Hons.) in electrical engineering and the M.Sc. degree (Hons.) in information technology from Technische Universität München (TUM), Germany, in 2013 and 2016, respectively. He is currently pursuing the Ph.D. degree with the Coding for Communications and Data Storage (COD) Group, TUM, where he is involved in research about coding theory for insertion and deletion errors and modern data storage systems. His research interests include parameter estimation, communications, and circuit theory. In 2019, he received the Memorable Paper Award from the Non-Volatile Memories Workshop.

Antonia Wachter-Zeh (Senior Member, IEEE) received the M.Sc. degree in communications technology from Ulm University, Germany, in 2009, and the Ph.D. degree from Ulm University and the Universite de Rennes 1, Rennes, France, in 2013. From 2013 to 2016, she was a Post-Doctoral Researcher at the Technion—Israel Institute of Technology, Haifa, Israel. From 2016 to 2020, she was a Tenure Track Assistant Professor at TUM. She is currently an Associate Professor with the Department of Electrical and Computer Engineering, Technical University of Munich (TUM), Munich, Germany. Her research interests include coding theory, cryptography, and information theory and their application to storage, communications, privacy, and security. She was a recipient of the DFG Heinz Maier-Leibnitz-Preis and an ERC Starting Grant. She is currently an Associate Editor of the IEEE TRANSACTIONS ON INFORMATION THEORY.