Reconstruction of Sequences in DNA Storage

Maria Abu Sini
Technion - Israel Institute of Technology
Haifa 3200009, Israel
maria.as@cs.technion.ac.il

Eitan Yaakobi
Technion - Israel Institute of Technology
Haifa 3200009, Israel
yaakobi@cs.technion.ac.il

***Abstract*—The *sequence reconstruction problem* corresponds to a model in which a sequence from some code is transmitted over several noisy channels. The channels are almost independent as it is only required that their outputs are different. The main problem under this paradigm is to determine the minimum number of channels required to reconstruct the transmitted sequence. This problem is equivalent to finding the maximum intersection size between two balls of any possible two inputs, where the balls are all possible channel outputs. Motivated by the error behavior in the DNA storage channel, this work extends this study to the case where the channels are prone to substitutions, insertions, and deletions. For the case of only substitutions, we also present a decoder of optimal complexity, which improves upon a recent construction of such a decoder. Lastly, it is also studied how the decoder is simplified in case there are more channels than the minimum required number.**

I. Introduction

As a result of the constant increase in the information generated these days and since energy conservation has become an important issue, the storage community is actively searching for an innovative medium to store archival data, which composes most of the existing data in the world. One promising solution to fulfill these requirements is the application of DNA molecules to store digital information [3]. DNA is an attractive solution for large-scale archival storage systems due to its unique attributes of durability, endurance, and extremely high density. Thanks to recent outstanding advancements in biotechnological applications, it is now possible to store archival data in synthetic DNA, and read it using sequencing techniques. Several experiments have been conducted during the past decade. The first ones are the implementations of Church et al. and Goldman et al.; the first managed to recover 643 KB [2] while the second recovered 739 KB [4]. However, due to errors occurring at the stages of synthesis and sequencing, in both of these experiments, parts of the data were lost. More experiments and examples can be found in [5], [6], [8], [14], [19]–[21].

DNA synthesis refers to the process of artificially generating DNA strands that can store any digital data and DNA sequencing is the process of reading these strands back. Current synthesis technology does not generate a single strand of a requested sequence, but thousands to millions copies for each strand. All of these copies are stored in a DNA pool and a portion of them will be sequenced when retrieving the information. Moreover, sequencing is usually preceded by PCR (polymerase chain reaction) amplification which generates copies of a strand. As a result, when sequencing DNA strands, each DNA strand is read many times. First, because thousands of copies were synthesized, and second, because of the copies that were duplicated using PCR.

Since the processes of synthesis and sequencing are prone to errors, at the end of the sequencing process, there are many noisy copies of every strand that we intended to store. Since these copies cannot be easily differentiated, clustering algorithms are needed in order to distribute the noisy copies into several sets, such that each set consists of copies that are estimated to originate from the same sequence. Clustering algorithm and coding techniques were recently studied in [7], [15], [17]. In this paper, we assume that clustering is already done and thus we focus on the reconstruction step aiming to output the original strand or a strand with minimum number of errors. Thus, the sequencing's output followed by clustering is a set of clusters. Each cluster consists of sequencing reads that are estimated to be noisy copies of a specific strand, and the copies in this cluster are used in order to recover the original strand. The problem of using a set of erroneous sequences in order to recover the correct one falls under the definition of the *reconstruction problem* that was introduced by Levenshtein in [11], [12], and is the main topic of this paper.

The reconstruction model studied by Levenshtein and later by others is combinatorial. It is assumed that the information is a codeword in some code and it has several noisy copies. Then, the goal is to find the minimum number of noisy copies that guarantees unique decoding in the worst case. This number has to be larger than the largest intersection of two balls of any two codewords in the code, where the ball is the set of all noisy copies that can be received. Levenshtein studied the cases of substitution errors, the Johnson graphs, and several more general metric distances. Levenshtein's results for deletions and insertions in [12] were extended in [16] for insertions and in [9] for deletions. In [18], the connection between the reconstruction problem and associative memories was studied.

The rest of the paper is organized as follows. Section II presents the formal definition of the reconstruction problem. Motivated by the error behavior in DNA-based storage [10], we extend this problem to the case of substitutions, insertions, and deletions. In Section III, we take first steps in exploring this extension and study the case of single substitution and single insertion. Next, in Section IV, a decoding algorithm to the reconstruction problem with substitution errors of optimal complexity is presented. Lastly, in Section V, it is studied how the decoding algorithms can be simplified in case there are more copies than the minimum one that was studied by Levenshtein. Due to the lack of space, some of the proofs in the paper are omitted.

II. Definitions and Preliminaries

In this section we review the reconstruction problem which was first proposed by Levenshtein [11], [12]. Let V be the space of all possible words. Assume that there are several identical channels, such that for a transmitted word $\boldsymbol{w} \in V$, the output of every channel is some word in the error ball surrounding $\boldsymbol{w}$ and denoted by $B(\boldsymbol{w})$. Furthermore, assume that all channels' outputs are distinct. In this study it is assumed that the errors can be of several types and do not necessarily correspond to a specific distance metric; thus we keep the definition of the error ball as general as possible. For example, motivated by DNA-based storage systems, it will be assumed that every channel is prone to combinations of substitutions, deletions, and insertions. Then, the goal is to find the minimum number of channels needed so that the transmitted word can be determined definitely, i.e., in the worst case.

Assume the transmitted word $\boldsymbol{w}$ belongs to some code $\mathcal{C} \subseteq V$. The problem of finding the largest intersection

$$\max_{\boldsymbol{w}_1,\boldsymbol{w}_2\in\mathcal{C},\boldsymbol{w}_1\neq\boldsymbol{w}_2} \{|B(\boldsymbol{w}_1)\cap B(\boldsymbol{w}_2)|\}, \tag{1}$$

was first initiated and studied by Levenshtein [11], [12] and is referred here as the *reconstruction problem*. Levenshtein proved that the required minimum number of channels for the existence of successful decoder in the worst case has to be strictly larger than the value in (1).

In case the channels are prone only to substitutions, then Levenshtein has solved in [11] the problem for any length-n code $\mathcal{C} \subseteq \{0,1\}^n$ of minimum Hamming distance d. In other words, let $B_t^S(\boldsymbol{w})$ be the radius-t ball resulting from at most t substitutions to a word $\boldsymbol{w}$, then Levenshtein found the value

$$N_n^S(t,d) \triangleq \max_{\boldsymbol{w}_1,\boldsymbol{w}_2\in\mathcal{C},\boldsymbol{w}_1\neq\boldsymbol{w}_2} \left\{|B_t^S(\boldsymbol{w}_1)\cap B_t^S(\boldsymbol{w}_2)|\right\},$$

where $\mathcal{C}$ is any length-n code of minimum Hamming distance d. Since this value does not depend on the specific choice of the code $\mathcal{C}$, but only on its minimum Hamming distance, it was shown in [11] that

$$\begin{aligned} N_n^S(t,d) &= \max_{\boldsymbol{w}_1,\boldsymbol{w}_2\in\{0,1\}^n, d_H(\boldsymbol{w}_1,\boldsymbol{w}_2)\geqslant d} \left\{|B_t^S(\boldsymbol{w}_1)\cap B_t^S(\boldsymbol{w}_2)|\right\}, \\ &= \sum_{i=0}^{t-\lceil\frac{d}{2}\rceil} \binom{n-d}{i} \sum_{h=d-t+i}^{t-i} \binom{d}{h}. \end{aligned}$$

This problem is naturally extended for deletions and insertions. Here we use the *edit distance* between two words $\boldsymbol{w}_1, \boldsymbol{w}_2 \in \{0,1\}^n$ which is half of the minimum number of insertions and deletions required to convert $\boldsymbol{w}_1$ to $\boldsymbol{w}_2$. Furthermore, let $B_t^I(\boldsymbol{w})$, $B_t^D(\boldsymbol{w})$ be the radius-t ball that results from t insertions, deletions, to the word $\boldsymbol{w}$, respectively. Then, the reconstruction problem in this case translates to finding

$$N_n^D(t,d) \triangleq \max_{\boldsymbol{w}_1,\boldsymbol{w}_2\in\{0,1\}^n, d_E(\boldsymbol{w}_1,\boldsymbol{w}_2)\geqslant d} \left\{|B_t^D(\boldsymbol{w}_1)\cap B_t^D(\boldsymbol{w}_2)|\right\},$$

$$N_n^I(t,d) \triangleq \max_{\boldsymbol{w}_1,\boldsymbol{w}_2\in\{0,1\}^n, d_E(\boldsymbol{w}_1,\boldsymbol{w}_2)d} \left\{|B_t^I(\boldsymbol{w}_1)\cap B_t^I(\boldsymbol{w}_2)|\right\}.$$

The values of $N_n^D(t,1)$ and $N_n^I(t,1)$ were found by Levenshtein in [12]. Later, his results were extended in [16] for determining the value of $N_n^I(t,d)$ for all d and t and in [9] for the value $N_n^D(t,2)$ for all t.

These results are applicable for DNA-based storage systems. Each cluster may be used to recover the original strand as it consists of erroneous sequences of the same strand. Thus, we may consider the noisy copies of a cluster as channels in which the strand is transmitted. Under the restriction that each cluster suffers from only one type of errors, the results from [9], [11], [12], [16] determine the minimum size of a cluster that guarantees successful decoding for almost all cases. However, these results do not consider combinations of errors that can occur in DNA-based storage systems [10]. Therefore, this paper expands existing results and takes first steps in order to find the minimum number of channels required for decoding in the presence of combinations of errors.

For a word $\boldsymbol{w} \in \{0,1\}^n$, the ball $B_{t_1,t_2,t_3}(\boldsymbol{w})$ is the set of all possible words which result from at most t_1 substitutions and exactly t_2 deletions and t_3 insertions to $\boldsymbol{w}$. The following problem formulates the reconstruction problem for the case of substitutions, deletions, and insertions.

Problem 1. *Given a code $\mathcal{C} \subseteq \{0,1\}^n$, and $t_1, t_2, t_3 \in \mathbb{N}$, find the value of*

$$\max_{\boldsymbol{w}_1,\boldsymbol{w}_2\in\mathcal{C},\boldsymbol{w}_1\neq\boldsymbol{w}_2} \{|B_{t_1,t_2,t_3}(\boldsymbol{w}_1)\cap B_{t_1,t_2,t_3}(\boldsymbol{w}_2)|\}. \tag{2}$$

For $\mathcal{C} = \{0,1\}^n$ the value in (2) is denoted by $N_n(t_1,t_2,t_3)$.

Before we proceed to study Problem 1, we note that, to the best of our knowledge, finding the size of the ball $B_{t_1,t_2,t_3}(\boldsymbol{w})$ has not been studied before and it is a challenging problem by itself. It is well-known that the deletions balls are not regular, that is, the size of the deletion ball $B_t^D(\boldsymbol{w})$ depends on the choice of the word $\boldsymbol{w}$, where the substitutions and insertions balls are regular. Hence, while it is possible to see that the case of single-deletion and single-substitution is not regular, it was surprising to find out that the single-insertion and single-substitution case is not regular either, as will be shown next.

For a word $\boldsymbol{w} \in \{0,1\}^n$, its number of runs is denoted by $r(\boldsymbol{w})$, and its *runs-profile* is the length-$r(\boldsymbol{w})$ vector $\ell(\boldsymbol{w}) = (\ell_1,\ldots,\ell_{r(\boldsymbol{w})})$, which specifies the length of each of the $r(\boldsymbol{w})$ runs in $\boldsymbol{w}$. The runs-profile determines the word $\boldsymbol{w}$ up to its complement.

Theorem 2. *Let $\boldsymbol{w} \in \{0,1\}^n$ be a word with runs-profile vector $\ell(\boldsymbol{w}) = (\ell_1,\ldots,\ell_{r(\boldsymbol{w})})$. The following properties hold:*

1) *The single-deletion single-substitution ball size of $\boldsymbol{w}$ is*

$$|B_{1,1,0}(\boldsymbol{w})| = (n-3)r(\boldsymbol{w}) + 4,$$

for $r(\boldsymbol{w}) \geqslant 2$ and $|B_{1,1,0}(\boldsymbol{w})| = n$ if $r(\boldsymbol{w}) = 1$.

2) *The single-insertion single-substitution ball size of $\boldsymbol{w}$ is*

$$|B_{1,0,1}(\boldsymbol{w})| = (n+2)^2 - 2 - \sum_{i=1}^{r(\boldsymbol{w})} \frac{\ell_i(\ell_i+5)}{2}.$$

Proof: We prove the first part of the lemma for the single-deletion single-substitution case. In case $r(\boldsymbol{w}) = 1, 2$ the property can be verified, so it is assumed for the rest of the proof that $r(\boldsymbol{w}) \geqslant 3$. Notice also that there is no difference if there is first a deletion and then a substitution or the opposite, so we consider the first case. For $i, 1 \leqslant i \leqslant r(\boldsymbol{w})$, let $\boldsymbol{w}^i$ be the word that results from the deletion of a bit in the i-th run of $\boldsymbol{w}$. Thus, it holds that

$$B_{1,1,0}(\boldsymbol{w}) = \bigcup_{\boldsymbol{w}'\in B_{0,1,0}(\boldsymbol{w})} B_{1,0,0}(\boldsymbol{w}') = \bigcup_{1\leqslant i\leqslant r(\boldsymbol{w})} B_{1,0,0}(\boldsymbol{w}^i).$$

The size of $B_{1,1,0}(\boldsymbol{w})$ is determined by the following cases.

1) For $1 \leqslant i \leqslant r(\boldsymbol{w})$, $|B_{1,0,0}(\boldsymbol{w}^i)| = n$.
2) For $2 \leqslant i \leqslant r(\boldsymbol{w})$, $d_H(\boldsymbol{w}^{i-1},\boldsymbol{w}^i) = 1$ and thus $|B_{1,0,0}(\boldsymbol{w}^{i-1}) \cap B_{1,0,0}(\boldsymbol{w}^i)| = 2$.
3) For $3 \leqslant i \leqslant r(\boldsymbol{w})$, $d_H(\boldsymbol{w}^{i-2},\boldsymbol{w}^i) = 2$ and thus $|B_{1,0,0}(\boldsymbol{w}^{i-1}) \cap B_{1,0,0}(\boldsymbol{w}^i)| = 2$.
4) For $3 \leqslant i \leqslant r(\boldsymbol{w})$, $|B_{1,0,0}(\boldsymbol{w}^{i-2}) \cap B_{1,0,0}(\boldsymbol{w}^{i-1}) \cap B_{1,0,0}(\boldsymbol{w}^i)| = 1$.
5) For all $3 \leqslant j$ and $j+1 \leqslant i \leqslant r(\boldsymbol{w})$, $d_H(\boldsymbol{w}^{i-j},\boldsymbol{w}^i) \geqslant 3$ and thus $|B_{1,0,0}(\boldsymbol{w}^{i-j}) \cap B_{1,0,0}(\boldsymbol{w}^i)| = 0$.

Finally, the size of the ball $B_{1,1,0}(\boldsymbol{w})$ is calculated according to the inclusion-exclusion principle to be

$$\begin{aligned} |B_{1,1,0}(\boldsymbol{w})| &= r(\boldsymbol{w})\cdot n - (2r(\boldsymbol{w})-3)\cdot 2 + (r(\boldsymbol{w})-2)\cdot 1 \\ &= (n-3)r(\boldsymbol{w}) + 4. \end{aligned}$$

■

According to Theorem 2, it follows that the maximum size of the single-substitution single-insertion ball is n^2+n+2, which is achieved only for the alternating words, that is, $0101\cdots$ or $1010\cdots$. Similarly, the maximum size of the single-substitution single-deletion ball is n^2-3n+4.

The method from Theorem 2 for finding the size of the ball $B_{1,1,0}(\boldsymbol{w})$ uses the observation that for all $1 \leqslant i_1 < i_2 \leqslant r(\boldsymbol{w})$, it holds that $d_H(\boldsymbol{w}^{i_1},\boldsymbol{w}^{i_2}) = i_2 - i_1$, and then the size of the ball is calculated according to the inclusion-exclusion principle. This method can be used also to find the size of the set $B_{t_1,1,0}(\boldsymbol{w})$ for other values of t_1, however even for $t_1 = 2$ this calculation becomes significantly cumbersome.

III. The Single-Substitution Single-Insertion Case

In this section we make first steps towards studying Problem 1 and the value $N_n(t_1,t_2,t_3)$. Our focus here will be on the case study of $N_n(1,0,1)$, in which we show that $N_n(1,0,1) = \frac{n^2}{4} + O(n)$. While the largest intersection in the case of only substitution, deletions, or insertions, was received for words which differ on any single bit or variations of the alternating words, here we notice a special behavior where the largest intersection is received for words which differ in their middle bit. Namely, the following lemma assures this lower bound on $N_n(1,0,1)$.

Lemma 3. *For $4 \leqslant n$ even, $N_n(1,0,1) \geqslant \frac{n^2}{4} + 3n + 1$.*

Proof: This result follows from the intersection of the balls of the words $\boldsymbol{w}_1 = 0^{\frac{n-2}{2}}10^{\frac{n-2}{2}}$ and $\boldsymbol{w}_2 = 0^{\frac{n-1}{2}}01^{\frac{n-2}{2}}$, while the notation b^m for $b \in \{0,1\}$ and a positive integer m corresponds to the concatenation of the bit b m times. Thus, it can be proved that

$$|B_{1,0,1}(\boldsymbol{w}_1) \cap B_{1,0,1}(\boldsymbol{w}_2)| = \frac{n^2}{4} + 3n + 1.$$

■

Theorem 4. *It holds that $N_n(1,0,1) = \frac{n^2}{4} + O(n)$.*

Proof: We consider the intersection $B_{1,0,1}(\boldsymbol{w}) \cap B_{1,0,1}(\boldsymbol{w}')$ of two words $\boldsymbol{w},\boldsymbol{w}' \in \{0,1\}^n$ such that $d_H(\boldsymbol{w},\boldsymbol{w}') \geqslant 2$, while the case of $d_H(\boldsymbol{w},\boldsymbol{w}') = 1$ can be handled similarly. Let $i, i+\ell-1$ be the first, last index in which $\boldsymbol{w}$ and $\boldsymbol{w}'$ differ, respectively. It holds that

$$B_{1,0,1}(\boldsymbol{w}) \cap B_{1,0,1}(\boldsymbol{w}') = \bigcup_{\boldsymbol{z}\in B_{0,0,1}(\boldsymbol{w}),\boldsymbol{z}'\in B_{0,0,1}(\boldsymbol{w}')} B_{1,0,0}(\boldsymbol{z}) \cap B_{1,0,0}(\boldsymbol{z}').$$

In order to bound the size $|B_{1,0,1}(\boldsymbol{w}) \cap B_{1,0,1}(\boldsymbol{w}')|$ from above, we bound the size of $B_{1,0,0}(\boldsymbol{z}) \cap B_{1,0,0}(\boldsymbol{z}')$ for all $\boldsymbol{z} \in B_{0,0,1}(\boldsymbol{w})$ and $\boldsymbol{z}' \in B_{0,0,1}(\boldsymbol{w}')$. Let $\boldsymbol{z}(\alpha,m_1), \boldsymbol{z}'(\beta,m_2)$ be the word that results from inserting the bit $\alpha,\beta \in \{0,1\}$ right before the m_1-th, m_2-th value in $\boldsymbol{w},\boldsymbol{w}'$ (or at the end for $m_1 = n+1, m_2 = n+1$), respectively. We study the intersection size

$$|B_{1,0,0}(\boldsymbol{z}(\alpha,m_1)) \cap B_{1,0,0}(\boldsymbol{z}'(\beta,m_2))|$$

for all possible values of α,β,m_1,m_2. Since we study intersections of at most a single substitution, we only need to consider the cases in which $d_H(\boldsymbol{z}(\alpha,m_1),\boldsymbol{z}'(\beta,m_2)) \leqslant 2$. For shorthand, in the following cases $\boldsymbol{z},\boldsymbol{z}'$ refers to $\boldsymbol{z}(\alpha,m_1), \boldsymbol{z}'(\beta,m_2)$, respectively.

1) $m_1,m_2 \in [1,i]$ or $m_1,m_2 \in [i+\ell,n+1]$ or $m_1,m_2 \in [i+1,i+\ell-1]$. If $m_1 \neq m_2$ then $d_H(\boldsymbol{z},\boldsymbol{z}') \leqslant 2$ only if α and β are inserted to the same run and are equal to the bits in the run. Otherwise, $m_1 = m_2$ and in this case $d_H(\boldsymbol{z},\boldsymbol{z}') \leqslant 2$ only if $\alpha = \beta$. In both cases the number of options for z and z' that satisfy $d_H(\boldsymbol{z},\boldsymbol{z}') \leqslant 2$ is $O(n)$ which implies that the total number of words in $B_{1,0,1}(\boldsymbol{w}) \cap B_{1,0,1}(\boldsymbol{w}')$ of this form in $O(n)$.
2) $m_1 \in [1,i], m_2 \in [i+1,i+\ell-1]$. We observe that $d_H(\boldsymbol{z},\boldsymbol{z}') \leqslant 2$, only if α is inserted to one of the two runs that precede w_i and one of the followings holds:
   - $\alpha = w'_{m_1}$ and $\beta = w_{m_2-1}$. Since α is inserted to a limited number of runs and is equal to w'_{m_1}, then there are at most three possible options for $\boldsymbol{z}$. There are at most ℓ possible options for $\boldsymbol{z}'$. Each pair of $\boldsymbol{z}$ and $\boldsymbol{z}'$ is of Hamming distance at least 1. Thus, $|B_{1,0,0}(\boldsymbol{z}) \cap B_{1,0,0}(\boldsymbol{z}')| \leqslant 2$.
   - $\alpha \neq w'_{m_1}$ and $\beta = w_{m_2-1}$. In this case $\boldsymbol{z}$ and $\boldsymbol{z}'$ that satisfy $d_H(\boldsymbol{z},\boldsymbol{z}') \leqslant 2$ differ only in the pairs of bits α, w'_{m_1} and $w_{i+\ell-1}, w'_{i+\ell-1}$.
   - $\alpha = w'_{m_1}$ and $\beta \neq w_{m_2-1}$. In this case $\boldsymbol{z}$ and $\boldsymbol{z}'$ that satisfy $d_H(\boldsymbol{z},\boldsymbol{z}') \leqslant 2$ differ only in the pairs of bits β, w_{m_2-1} and $w_{i+\ell-1}, w'_{i+\ell-1}$.

   The number of words that can be generated in $B_{1,0,1}(\boldsymbol{w}) \cap B_{1,0,1}(\boldsymbol{w}')$ in the first, second, third case is $O(n), O(i), O(\ell)$, respectively. The cases of $m_1 \in [i+1,i+\ell-1], m_2 \in [1,i]$, $m_1 \in [i+\ell,n+1], m_2 \in [i+1,i+\ell-1]$, and $m_1 \in [i+1,i+\ell-1], m_2 \in [i+\ell,n+1]$ are proved in a similar way so the total number of words that can be generated in $B_{1,0,1}(\boldsymbol{w}) \cap B_{1,0,1}(\boldsymbol{w}')$ in all of these cases is $O(n)$.
3) $m_1 \in [1,i], m_2 \in [i+\ell,n+1]$. It holds that $d_H(\boldsymbol{z},\boldsymbol{z}') \leqslant 2$ only if α is inserted in one of the three runs that precede w_i, and β is inserted in one of the three following $w'_{i+\ell-1}$. Therefore, if $\alpha = w'_{m_1}$ or $\beta = w_{m_2-1}$, then the number of generated words in $B_{1,0,1}(\boldsymbol{w}) \cap B_{1,0,1}(\boldsymbol{w}')$ is $O(n)$. Next, assume $\alpha \neq w'_{m_1}$, $\beta \neq w_{m_2-1}$. Assume also $m_1 < i$ and $m_2 > i+\ell$ as for $m_1 = i$ or $m_2 = i+\ell-1$, $O(n)$ words can be generated. We observe that for such α,β,m_1 and m_2, that satisfy $d_H(\boldsymbol{z},\boldsymbol{z}') = 2$, two words can be found in $B_{1,0,0}(\boldsymbol{z}) \cap B_{1,0,0}(\boldsymbol{z}')$ where for the first one, denoted by $\boldsymbol{y}_1$, α is in the m_1-th position and β is in the m_2-th position, while for the other, denoted by $\boldsymbol{y}_2$, w_{m_1} in the m_1-th position and w_{m_2-1} in the m_2-th position. $\boldsymbol{y}_1$ was not included in the previous cases while $\boldsymbol{y}_2$, as illustrated in the table below, was included, specifically by $\boldsymbol{z}_1, \boldsymbol{z}'_2$ that result from inserting $\alpha = w_{m_1}$ before w_{m_1}, $\beta = w_{i+\ell-2}$ before $w'_{i+\ell-1}$, respectively. Thus, such insertions generate $(i+2)(n-(i+\ell-1)+2)+O(n)$ words in $B_{1,0,1}(\boldsymbol{w}) \cap B_{1,0,1}(\boldsymbol{w}')$. Note that in case $w_{i+\ell} \neq w_{i+\ell-1}$ or $w_{i-1} \neq w'_i$ then at most $O(n)$ words are generated in $B_{1,0,1}(\boldsymbol{w}) \cap B_{1,0,1}(\boldsymbol{w}')$.

Index	$\boldsymbol{z}$	$\boldsymbol{z}'$	$\boldsymbol{y}_1$	$\boldsymbol{y}_2$	$\boldsymbol{z}_1$	$\boldsymbol{z}'_2$
1	w_1	w_1	w_1	w_1	w_1	w_1
2	w_2	w_2	w_2	w_2	w_2	w_2
⋮	⋮	⋮	⋮	⋮	⋮	⋮
m_1	α	w_{m_1}	α	w_{m_1}	$\alpha{=}w_{m_1}$	w_{m_1}
$m_1{+}1$	w_{m_1}	w_{m_1+1}	w_{m_1}	w_{m_1}	w_{m_1}	w_{m_1+1}
⋮	⋮	⋮	⋮	⋮	⋮	⋮
i	w_{i-1}	w'_i	w_{i-1}	w_{i-1}	w_{i-1}	w'_i
$i{+}1$	w_i	w'_{i+1}	w_i	w_i	w_i	w'_{i+1}
⋮	⋮	⋮	⋮	⋮	⋮	⋮
$i{+}\ell{-}1$	$w_{i+\ell-2}$	$w'_{i+\ell-1}$	$w_{i+\ell-2}$	$w_{i+\ell-2}$	$w_{i+\ell-2}$	$\beta{=}w_{i+\ell-2}$
$i{+}\ell$	$w_{i+\ell-1}$	$w_{i+\ell}$	$w_{i+\ell-1}$	$w_{i+\ell-1}$	$w_{i+\ell-1}$	$w'_{i+\ell-1}$
⋮	⋮	⋮	⋮	⋮	⋮	⋮
m_2	w_{m_2-1}	β	β	w_{m_2-1}	w_{m_2-1}	w_{m_2-1}
⋮	⋮	⋮	⋮	⋮	⋮	⋮
n	w_{n-1}	w_{n-1} t	w_{n-1}	w_{n-1}	w_{n-1}	w_{n-1}
$n{+}1$	w_n	w_n	w_n	w_n	w_n	w_n

4) $m_1 \in [i+\ell,n+1], m_2 \in [1,i]$. By the same analysis of case 3 it can be proved that for $\boldsymbol{w},\boldsymbol{w}'$ such that $w_{i+\ell} = w'_{i+\ell-1}$ and $w_{i-1} = w_i$ at most $(i+2)(n-(i+\ell-1)+2)+O(n)$ words are generated and for other cases of $\boldsymbol{w},\boldsymbol{w}'$, only $O(n)$ words are generated.

To conclude, only one of the fourth and fifth cases can generate $(i+2)(n-(i+\ell-1)+2)$ words in $B_{1,0,1}(\boldsymbol{w}) \cap B_{1,0,1}(\boldsymbol{w}')$ and all other cases $O(n)$, which confirms the lemma.

■

IV. Decoder for Substitution Errors

In this section, it is studied how to construct a decoder for the reconstruction problem in the substitutions case with optimal complexity. Since $N_n^S(t,d)+1=\Theta(n^{t-\lceil\frac{d}{2}\rceil})$, the order of magnitude of the number of bits in any subset $Y\subseteq B_t^S(x)$ of size $N_n^S(t,d)+1$ is $\Theta(n^{t-\lceil\frac{d}{2}\rceil+1})$. Hence, the complexity order of any decoder is at least this value, and a decoder achieving this complexity will be called *optimal*. A decoder for this problem was presented in [18] for all d and t, however its complexity is $\Theta(n^{2t-d})$, and only the case where $d=3$ has been improved in [18] to have an optimal decoder, i.e., with complexity $\Theta(n^{t-1})$. In this section we show how to construct an optimal decoder for all d and t. For the rest of the paper it is assumed that d and t are fixed positive integer, n is large enough, and $t>(d-1)/2+1$ (the case $t=(d-1)/2+1$ has been solved in [18]). Furthermore, since for every odd d, $N_n^S(t,d)=N_n^S(t,d+1)$ [18], it is also assumed that d is an odd integer. For shorthand, $N_n^S(t,d)+1$ is denoted by $N_{t,d}$.

A set $Y=\{\boldsymbol{y}_1,\ldots,\boldsymbol{y}_N\}\subseteq\{0,1\}^n$ is said to be decoded according to the majority algorithm with threshold τ and we denote $\boldsymbol{z}=\mathrm{maj}_\tau(Y)$, where $\boldsymbol{z}$ is the algorithm's output word, if the following holds: For all $1\leqslant i\leqslant n$, let

$$m_{i,0}=|\{j: j\in[N], y_{j,i}=0\}|, m_{i,1}=|\{j: j\in[N], y_{j,i}=1\}|.$$

If $|m_{i,0}-m_{i,1}|\leqslant\tau$ then, $z_i=?$. Otherwise, if $m_{i,0}>m_{i,1}$ then $z_i=0$ and if $m_{i,0}<m_{i,1}$ then $z_i=1$.

For a code $\mathcal{C}\subseteq\{0,1\}^n$ of minimum Hamming distance d, we assume that it has a complete decoder $\mathcal{D}_\mathcal{C}$ that can successfully correct at most $\frac{d-1}{2}$ errors. If the number of errors is greater than this value there is no guarantee on the decoder's success.

In the algorithm, the following value $\tau_{t,d}$ will be used for the threshold of the majority algorithm

$$\tau_{t,d}\triangleq\frac{4}{d+1}\sum_{i=0}^{t-\frac{d+1}{2}}\binom{n-\frac{d+1}{2}}{i}+\frac{d-3}{d+1}N_{t,d}.$$

It is possible to verify that $\tau_{t,d}<N_{t,d}$. For an integer t and a code $\mathcal{C}\subseteq\{0,1\}^n$ of minimum Hamming distance d, we define an algorithm that recovers the transmitted codeword denoted by $\boldsymbol{c}$ using a subset $Y\subseteq B_t^S(\boldsymbol{c})$ of size $N_{t,d}$.

Algorithm 5. The input to the decoder is a set of all $N_{t,d}$ channel outputs $Y=\{\boldsymbol{y}_1,\ldots,\boldsymbol{y}_{N_{t,d}}\}\subseteq B_t^S(\boldsymbol{c})$ for some $\boldsymbol{c}\in\mathcal{C}$ and it returns an estimation $\widehat{\boldsymbol{c}}$ on $\boldsymbol{c}$.
Step 1. $\boldsymbol{z}=\mathrm{maj}_{\tau_{t,d}}(Y)$.
Step 2. $S=\{i: i\in[n], z_i=?\}$.
Step 3. $Z=\{\boldsymbol{u}\in\{0,1\}^n: u_i=z_i \text{ for all } i\notin S\}$.
Step 4. For all $\boldsymbol{u}\in Z$, $\widehat{\boldsymbol{c}}=\mathcal{D}_\mathcal{C}(\boldsymbol{u})$. If $Y\subseteq B_t(\widehat{\boldsymbol{c}})$, output $\widehat{\boldsymbol{c}}$.

The correctness of the algorithm is proved using the next two lemmas. For $1\leqslant i\leqslant n$, let e_i be $e_i=|\{\boldsymbol{y}\in Y: y_i\neq c_i\}|$, i.e. the number of words in Y in which there is an error in the i-th position. It holds that, $z_i=?$ if $\frac{N_{t,d}-\tau_{t,d}}{2}\leqslant e_i\leqslant\frac{N_{t,d}+\tau_{t,d}}{2}$, and z_i is in error if $e_i>\frac{\tau_{t,d}+N_{t,d}}{2}$.

Lemma 6. *There are at most $\frac{d-1}{2}$ errors in the word $\boldsymbol{z}$ in Step 1.*

Proof: Assume in contrary that there is a set Y such that the word $\boldsymbol{z}$ generated in Step 1 contains at least $\frac{d+1}{2}$ errors. Assume without loss of generality that the first $\frac{d+1}{2}$ bits are erroneous bits. As specified above, for $1\leqslant i\leqslant\frac{d+1}{2}$

$$e_i\geqslant\frac{N_{t,d}+\tau_{t,d}}{2}+1=\frac{2}{d+1}\sum_{i=0}^{t-\frac{d+1}{2}}\binom{n-\frac{d+1}{2}}{i}+\frac{d-1}{d+1}N_{t,d}+1.$$

Therefore,

$$\begin{aligned}\sum_{i=1}^{\frac{d+1}{2}}e_i&\geqslant\sum_{i=0}^{t-\frac{d+1}{2}}\binom{n-\frac{d+1}{2}}{i}+\frac{d-1}{2}N_{t,d}+\frac{d+1}{2}\\&>\sum_{i=0}^{t-\frac{d+1}{2}}\binom{n-\frac{d+1}{2}}{i}+\frac{d-1}{2}N_{t,d}\\&=\frac{d+1}{2}\sum_{i=0}^{t-\frac{d+1}{2}}\binom{n-\frac{d+1}{2}}{i}+\frac{d-1}{2}\left(N_{t,d}-\sum_{i=0}^{t-\frac{d+1}{2}}\binom{n-\frac{d+1}{2}}{i}\right).\end{aligned}$$

On the other hand, there are at most $\sum_{i=0}^{t-\frac{d+1}{2}}\binom{n-\frac{d+1}{2}}{i}$ words in Y that can have erroneous values in all of the first $\frac{d+1}{2}$ positions. The other $N_{t,d}-\sum_{i=0}^{t-\frac{d+1}{2}}\binom{n-\frac{d+1}{2}}{i}$ words can have at most $\frac{d-1}{2}$ errors in the first $\frac{d+1}{2}$ positions. Therefore, the number of errors in these $\frac{d+1}{2}$ positions is upper bounded by

$$\sum_{i=1}^{\frac{d+1}{2}}e_i\leqslant\frac{d+1}{2}\sum_{i=0}^{t-\frac{d+1}{2}}\binom{n-\frac{d+1}{2}}{i}+\frac{d-1}{2}\left(N_{t,d}-\sum_{i=0}^{t-\frac{d+1}{2}}\binom{n-\frac{d+1}{2}}{i}\right),$$

which results with a contradiction. ∎

Lemma 7. *For n large enough it holds that $|S|\leqslant t\cdot\left(\frac{d+3}{2}\right)$.*

Proof: Since in each of the $N_{t,d}$ words in Y there are at most t errors, the total number of errors in all of the $N_{t,d}$ words is bounded by $tN_{t,d}$. Since every erasure requires at least $\frac{N_{t,d}-\tau_{t,d}}{2}$ errors in the copies of this bit, the maximum number of erasures in $\boldsymbol{z}$ is upper bounded by $\frac{tN_{t,d}}{\frac{N_{t,d}-\tau_{t,d}}{2}}=2t\frac{N_{t,d}}{N_{t,d}-\tau_{t,d}}$. The value $N_{t,d}$ satisfies

$$\begin{aligned}N_{t,d}&=\sum_{i=0}^{t-\frac{d+1}{2}}\binom{n-d}{i}\sum_{h=d-t+i}^{t-i}\binom{d}{h}+1\\&=\binom{d+1}{\frac{d+1}{2}}\binom{n-d}{t-\frac{d+1}{2}}+\Theta(n^{t-\frac{d+1}{2}-1}),\end{aligned}$$

and for n large enough we have that $N_{t,d}\geqslant\frac{d+3}{2}\sum_{i=0}^{t-\frac{d+1}{2}}\binom{n-\frac{d+1}{2}}{i}$. This implies that

$$\frac{N_{t,d}}{N_{t,d}-\tau_{t,d}}=\frac{N_{t,d}}{\frac{4}{d+1}\left(N_{t,d}-\sum_{i=0}^{t-\frac{d+1}{2}}\binom{n-\frac{d+1}{2}}{i}\right)}\leqslant\frac{d+3}{4},$$

and thus $|S|\leqslant t\cdot\left(\frac{d+3}{2}\right)$. ∎

Lastly, we conclude with the following theorem.

Theorem 8. *The output $\widehat{\boldsymbol{c}}$ of Algorithm 5 is the word $\boldsymbol{c}$. The algorithm's complexity is $\Theta(n^{t-\frac{d+1}{2}+1})$ and hence it is optimal.*

Proof: From Lemma 6, since there are at most $\frac{d-1}{2}$ errors in the word $\boldsymbol{z}$ of Step 1, one of the $2^{|S|}$ words in the set Z of Step 3 contains at most $\frac{d-1}{2}$ errors. Thus, the decoding of this word in Step 4 will give the correct word $\boldsymbol{c}$. We use here the result of Lemma 16 from [18], in which $\widehat{\boldsymbol{c}}=\boldsymbol{c}$ if and only if $Y\subseteq B_t(\widehat{\boldsymbol{c}})$.

The complexity of Step 1 is $\Theta(N_{t,d})=\Theta(n^{t-\frac{d+1}{2}+1})$. According to Lemma 7, the size of the sets S and Z is constant with respect to n. Thus, the decoding in Step 4 is invoked a constant number of times and the complexity of the condition in this step is $\Theta(n^{t-\frac{d+1}{2}+1})$. Together, we conclude that the algorithm's complexity is $\Theta(n^{t-\frac{d+1}{2}+1})$. We assumed here that the complexity of the decoder $\mathcal{D}_\mathcal{C}$ is $O(n^{t-\frac{d+1}{2}+1})$. ∎

V. Extensions for the Decoder

Typically, the number of reads for each strand in DNA-based storage system is different from the value found by Levenshtein and can not be directly controlled. Therefore, we study the case in which there are more reads than the minimum required one and show how to take advantage of them to construct a simpler decoder.

The simplest decoding algorithm one can think of is the majority decoder in which every bit is decoded by the majority of its copies, that is, $z = \mathrm{maj}_\tau(Y)$ for $\tau = 0$. Note that in case the size of Y is even, the algorithm outputs the erasure symbol ? for bits having the same number of one and zero estimations. In fact, this is the decoding algorithm Levenshtein presented for the case of $d = 1$ and any t, so the number of channels is $N_n^S(t,1)+1$. This number of channels is necessary for the success of the majority decoder even if the transmitted word belongs to a code of any minimum Hamming distance $d \geqslant 1$. However, in this case the majority decoder only needs to output a word with at most some $k \leqslant \lfloor \frac{d-1}{2} \rfloor$ errors, as these errors can be corrected by any decoder of the code. In the following theorem we present bounds on the required minimum number of channels such that the majority decoder outputs a word with at most k errors.

Theorem 9. *For $4 \leqslant k < t, n \in \mathbb{N}$ large enough, a word $\boldsymbol{w} \in \{0,1\}^n$, and a set $Y \subseteq B_t(\boldsymbol{w})$, if the size of Y is larger than*

$$2\sum_{j=\left\lceil \frac{k+1}{2} \right\rceil}^{t} \binom{k}{j-1} \sum_{i=0}^{t-j} \binom{n-k-1}{i} + 2\frac{\left\lceil \frac{k+1}{2} \right\rceil - 1}{k-2\left\lceil \frac{k+1}{2} \right\rceil + 3} \sum_{j=\left\lceil \frac{k+1}{2} \right\rceil}^{t} \left(\binom{k}{j-1} - \binom{k}{j} \right) \sum_{i=0}^{t-j} \binom{n-k-1}{i}$$

then the number of erasures and errors in $\mathrm{maj}_0(Y)$ is at most k. Moreover, there exists a set $Y \subseteq B_t(\boldsymbol{w})$ of size

$$2\sum_{j=\left\lceil \frac{k+1}{2} \right\rceil}^{t} \binom{k}{j-1} \sum_{i=0}^{t-j} \binom{n-k-1}{i} - 2\frac{k+1}{k-2\left\lceil \frac{k+1}{2} \right\rceil + 3} \left(\binom{k}{\left\lceil \frac{k+1}{2} \right\rceil - 2} \right) 2\left\lfloor \frac{\left\lceil \frac{k+1}{2} \right\rceil - 1}{k-2\left\lceil \frac{k+1}{2} \right\rceil + 3} \sum_{j=\left\lceil \frac{k+1}{2} \right\rceil}^{t} \left(\binom{k}{j-1} - \binom{k}{j} \right) \sum_{i=0}^{t-j} \binom{n-k-1}{i} \right\rfloor$$

such that there are at least $k+1$ erasures and errors in $\mathrm{maj}_0(Y)$.

As a direct result of Theorem 9, if d is odd, then for $k = \frac{d-1}{2}$, we get that the number of channels is $\Theta(n^{t-\lceil \frac{d+1}{4} \rceil})$, while the minimum number of channels by Levenshtein is $N_n(d,t)+1 = \Theta(n^{t-\frac{d-1}{2}})$. That is, the degree of n in the number of channels increases by $\frac{d-1}{2} - \lceil \frac{d+1}{4} \rceil = \lfloor \frac{d-3}{4} \rfloor$. However, the majority decoder does not have to be applied with $\tau = 0$, and as was done in Section IV it is possible to change the decoder's threshold τ. According to this approach the goal is to output a word with at most k_1 errors and k_2 erasures for any k_1 and k_2 such that $2k_1 + k_2 \leqslant d-1$. Then this word can be successfully decoded using the decoder of the code. We demonstrate this approach for $d = 3$.

The authors of [18] have suggested a decoder for a code with minimum distance 3 that is based on two steps and two thresholds, τ_1 and τ_2 where $\tau_1 < \tau_2$, and requires only the minimum number of channels, $N_n^S(3,t)+1$. The first step of their algorithm checks if $z_1 = \mathrm{maj}_{\tau_1}(Y)$ has some erasures, and if so then $z_2 = \mathrm{maj}_{\tau_2}(Y)$ is computed. It was proved that if the word z_1 has no erasures then it suffers from at most a single error. However, in case z_2 is computed then it might have at most $8t$ erasures, for n large enough. In order to deal with the erasures, all codewords that match on the erasures are examined aiming to find the one that contains the set Y in its ball. However, this approach increases the complexity of the decoder since it requires checking all possible codewords and for each one going over all of the words in Y.

However, if there are at least $N = 32\sum_{i=0}^{t-3}\binom{n-3}{i} + 13\binom{n-3}{t-2}$ channels, then the algorithm from [18] can be applied with thresholds $\tau_1 = 4\sum_{i=0}^{t-3}\binom{n-3}{i} + \binom{n-3}{t-2}$ and $\tau_2 = 8\sum_{i=0}^{t-3}\binom{n-3}{i} + 3\binom{n-3}{t-2}$ to ensure that the generated word has at most a single error or at most two erasures, which can then be corrected by the decoder of the code. This approach presents the tradeoff between the number of channels and the decoder's complexity by taking advantage of the large number of channels to improve the performance. Furthermore, in this case the order of the number of channels does not increase since for n large enough $N \approx \frac{13}{6} N_n(3,t)$.

References

[1] J. Bornholt, R. Lopez, D. M. Carmean, L. Ceze, G. Seelig, and K. Strauss, "A DNA-based archival storage system," *ASPLOS*, pp. 637–649, Atlanta, GA, Apr. 2016.

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

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

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

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

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

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

[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. Gabrys and E. Yaakobi, "Sequence reconstruction over the deletion channel," *IEEE Trans. on Inform. Theory*, vol. 64, no. 4, pp. 2924–2931, Apr. 2018.

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

[11] V. I. Levenshtein, "Efficient reconstruction of sequences," *IEEE Trans. on Inform. Theory*, vol. 47, no. 1, pp. 2–22, Jan. 2001.

[12] V. I. Levenshtein, "Efficient reconstruction of sequences from their subsequences or supersequences," *Journal of Combin. Theory, Ser. A*, vol. 93, no. 2, pp. 310–332, 2001.

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

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

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

[16] F. Sala, R. Gabrys, C. Schoeny, and L. Dolecek, "Exact reconstruction from insertions in synchronization codes," *IEEE Trans. on Inform. Theory*, vol. 63, no. 4, pp. 2428–2445, Jan. 2017.

[17] T. Shinkar, E. Yaakobi, A. Lenz, and A. Wachter-Zeh, "Clustering correcting codes," to appear *IEEE Int. Symp. Inf. Theory*, Paris, France, Jul. 2019.

[18] E. Yaakobi and J. Bruck, "On the uncertainty of information retrieval in associative memories," *IEEE Trans. on Inform. Theory*, vol. 65, no. 4, pp. 2155–2165, Apr. 2019.

[19] S. H. T. Yazdi, R. Gabrys, and O. Milenkovic, "Portable and error-free DNA-based data storage," *Cold Spring Harbor Labs Journals*, 2016.

[20] S. H. T. Yazdi, H. M. Kiah, E. Garcia-Ruiz, J. Ma, H. Zhao, and O. Milenkovic, "DNA-based storage: Trends and methods," *IEEE Trans. Mol., Bio. and Multi-Scale Comm.*, vol. 1, no. 3, pp. 230–248, 2015.

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