Reconstruction of Strings From Their Substrings Spectrum

Sagi Marcovich, *Student Member, IEEE*, and Eitan Yaakobi, *Senior Member, IEEE*

***Abstract*— This paper studies reconstruction of strings based upon their substrings spectrum. Under this paradigm, it is assumed that all substrings of some fixed length are received and the goal is to reconstruct the string. While many existing works assumed that substrings are received error free, we follow in this paper the noisy setup of this problem that was first studied by Gabrys and Milenkovic. The goal of this study is twofold. First we study the setup in which not all substrings in the multispectrum are received, and then we focus on the case where the read substrings are not error free. In each case we provide specific code constructions of strings that their reconstruction is guaranteed even in the presence of failure in either model. We present efficient encoding and decoding maps and analyze the cardinality of the code constructions, while studying the cases where the rates of our codes approach 1.**

***Index Terms*— Reconstruction of sequences, DNA sequencing, substring-unique strings, substring-distant strings.**

I. Introduction

IN MANY storage and communication channels it is not possible to receive the transmitted or stored string as one unit, even in its noisy version. Rather, the information about the string can only be provided in several other forms such as a list of its subsequences, statistics on its symbols, and more. This class of models usually falls under the general framework of the *string reconstruction problems*. There are several instances of this setup, such as the *k-deck problem* [11], [23], [29] and the *reconstruction from substring compositions problem* [1], [2], [7], [16], [25], [26], [30], [31]. Similar problems under this paradigm are the *trace reconstruction problem* [5] and the *reconstruction problem* by Levenshtein [19], however in these setups the string is received as one unit multiple times with possible errors.

This paper studies an important setup for this class of problems, where it is assumed that the information about the string is conveyed by the multispectrum of all its substrings of some fixed length. Under this paradigm, the goal is to reconstruct the string and the success of this process usually depends on the length of the read substrings and the stored string. This model of strings reconstruction is motivated by current DNA sequencing technologies and in particular shotgun DNA sequencing [21]. In this method, the DNA strand is broken into multiple fragments, called *reads*, which are then assembled together to reconstruct the strand [9], [22], [28].

Mathematically speaking, for a length-n string $\boldsymbol{w}$ and a positive integer L, its *L-multispectrum*, denoted by $S_L(\boldsymbol{w})$, is the multiset of all its length-L substrings, $S_L(\boldsymbol{w}) = \{\boldsymbol{w}_{1,L}, \boldsymbol{w}_{2,L}, \ldots, \boldsymbol{w}_{n-L+1,L}\}$, where $\boldsymbol{w}_{i,L}$ is the substring $(w_i, w_{i+1}, \ldots, w_{i+L-1})$. Then, the goal is to reconstruct the string $\boldsymbol{w}$ given its multispectrum $S_L(\boldsymbol{w})$. If a string can be uniquely reconstructed from its L-multispectrum, then it is called *L-reconstructible*. It was proved by Ukkonen [32] that if all length-$(L-1)$ substrings of $\boldsymbol{w}$ are different from each other, then the string $\boldsymbol{w}$ is L-reconstructible. A string $\boldsymbol{w}$ that satisfies this constraint is referred as *$(L-1)$-substring unique*. Based upon this property, it was recently proved in [12], [15] that if $L = \lceil a \log(n) \rceil$ for some fixed value of $a > 1$, then the asymptotic rate of all L-reconstructible strings approaches 1.

Several recent papers have taken an information-theoretic point of view to the string assembly problem. The goal of these works was to study the fundamental limits of reconstructing $\boldsymbol{w}$ from $S_L(\boldsymbol{w})$ with a fixed failure probability under different setups and various error models. Arratia *et al.* [4] studied the limits of any assembly algorithm that recovers $\boldsymbol{w}$ from $S_L(\boldsymbol{w})$ where $\boldsymbol{w}$ is an i.i.d DNA string, and later Motahari *et al.* [25] studied the case where only a subset of $S_L(\boldsymbol{w})$ is available, while each read begins at a uniformly distributed location of the string. They both showed that if the reads are long enough to have no repeats, then reconstruction is possible with high probability. This was then extended in [26] for the case where every read is transferred through a symmetric substitution noisy read channel and in [16] it was assumed that the reads are corrupted by at most some fixed number of edit errors. Moreover, it has been shown that if $\boldsymbol{w}$ satisfies several constraints, which are based on its repeats statistics, then it can be assembled with high probability from $S_L(\boldsymbol{w})$ [7] or a subset of it [30]. Another variation of the string assembly problem, which allows to partially reconstruct the string $\boldsymbol{w}$, was studied in [31].

In this paper, we follow the recent works by Gabrys and Milenkovic [15] and by Chang *et al.* [8] and assume that the L-multispectrum is not received error free, while still requiring to reconstruct $\boldsymbol{w}$ in the *worst case*. We consider two models of this setup. In the first one, it is assumed that not all substrings in the L-multispectrum were read so only a subset of $S_L(\boldsymbol{w})$ is received. The second model assumes that all reads in the

Manuscript received February 15, 2020; revised February 5, 2021; accepted May 20, 2021. Date of publication May 28, 2021; date of current version June 16, 2021. This article was presented in part at the 2020 IEEE International Symposium on Information Theory (ISIT) (reference [24]). *(Corresponding author: Eitan Yaakobi.)*

The authors are with the Department of Computer Science, Technion—Israel Institute of Technology, Haifa 3200003, Israel (e-mail: sagimar@cs.technion.ac.il; yaakobi@cs.technion.ac.il).

Communicated by A. Gohari, Associate Editor At Large.

Digital Object Identifier 10.1109/TIT.2021.3084864

L-multispectrum were received, however some of them might be erroneous. An important tool in our constructions uses the set of substring unique strings and we also study its extension. Namely, for fixed L and d, it is said that $\boldsymbol{w}$ is an (L, d)-*substring distant* string if the Hamming distance between any two of its length-L substrings is at least d. We study the cardinality of this set of strings and show an encoding and decoding maps for this constraint.

While the motivation of this problem originates from shotgun DNA sequencing where the fragments, i.e. substrings, of the DNA strands are read and should be used to reconstruct the DNA strand [21], the coding part does not match this model since natural DNA can be arbitrary and we do not necessarily have control on its data. However, we find the coding part to be important for DNA storage systems [6], [10], [17], [18], [33] where it is possible to encode the data and especially for future ones where long DNA strands will be able to be synthesized. Lastly, we consider in the paper only substitution errors, which were observed to be dominant errors in several previous DNA storage experiments [27], while the extension for edit errors is left for future work.

The rest of the paper is organized as follows. In Section II, we formally define the codes and constraints studied in this paper and review several previous results. In Section III, we study the case where an incomplete multispectrum is received. Section IV studies the setup where some of the read substrings are noisy as well as (L, d)-substring distant strings. Another construction for noisy substrings is presented in Section V. Finally, Section VI concludes the paper.

II. Definitions and Preliminaries

In this section we formally define the notations, codes, and constraints studied in the paper. For an integer $i \in \mathbb{N}$ we denote by $[i]$ the set $\{1, \ldots, i\}$. For a multiset A, let $|A|$ denote the number of elements in A (with repetitions). For a set I of integers and $a \in I$, $b_I(a)$ denotes the binary representation of the index of a in I using $\lceil \log(|I|) \rceil$ bits, when the integers are ordered in an increasing order. When I is omitted, it is implied that $I = [n]$, while n will be clear from the context.

Let $\Sigma = \{0, 1\}$ denote the binary alphabet, n an integer, and $\boldsymbol{w} \in \Sigma^n$ a string. For two positive integers i and k such that $i + k - 1 \leqslant n$, let $\boldsymbol{w}_{i,k}$ denote the length-k substring of $\boldsymbol{w}$ starting at position i. Additionally, let $\text{Pref}_k(\boldsymbol{w}) = \boldsymbol{w}_{1,k}$, $\text{Suff}_k(\boldsymbol{w}) = \boldsymbol{w}_{n+1-k,k}$ denote the *k-prefix*, *k-suffix* of $\boldsymbol{w}$, respectively. For two strings $\boldsymbol{w}, \boldsymbol{x} \in \Sigma^n$, $d_H(\boldsymbol{w}, \boldsymbol{x})$ is the Hamming distance between $\boldsymbol{w}$ and $\boldsymbol{x}$ and $w_H(\boldsymbol{w}) = d_H(\boldsymbol{w}, \boldsymbol{0})$ is the Hamming weight of $\boldsymbol{w}$. For a multiset $S = \{\boldsymbol{s}_1, \ldots, \boldsymbol{s}_m\} \subseteq \Sigma^n$ of strings, $d_H(S)$ is defined to be the *minimum Hamming distance of* S, which is the minimum Hamming distance among all pairs of strings in S, i.e., $d_H(S) = \min_{1 \leqslant i < j \leqslant m}\{d_H(\boldsymbol{s}_i, \boldsymbol{s}_j)\}$. For a nonnegative integer $r \leqslant n$, $\mathcal{B}_r(\boldsymbol{w})$ denotes the radius-r Hamming ball around $\boldsymbol{w}$, that is, $\mathcal{B}_r(\boldsymbol{w}) = \{\boldsymbol{x} \in \Sigma^n \mid d_H(\boldsymbol{w}, \boldsymbol{x}) \leqslant r\}$.

For a string $\boldsymbol{w} \in \Sigma^n$ and a positive integer $L \leqslant n$, the set $S_L(\boldsymbol{w})$ is defined to be the *L-multispectrum of* $\boldsymbol{w}$, which is the *multiset* of all its length-L substrings

$$S_L(\boldsymbol{w}) = \{\boldsymbol{w}_{1,L}, \boldsymbol{w}_{2,L}, \ldots, \boldsymbol{w}_{n-L+1,L}\}.$$

The main family of strings studied in this paper is defined in the next definition.

Definition 1: A string $\boldsymbol{w} \in \Sigma^n$ is called an (L, d)**-substring distant string** if the Hamming distance of its L-multispectrum is at least d, that is, $d_H(S_L(\boldsymbol{w})) \geqslant d$. For $d = 1$, we refer to an $(L, 1)$-substring distance string as an L**-substring unique string**.

We note that the case of $d = 1$ has also been studied in [12] and was referred as *repeat-free words*. Clearly, for given n and d, the number of (L, d)-substring distant strings grows as L increases. This follows from the observation that if $\boldsymbol{w}$ is (L, d)-substring distant, then it is immediately derived that $\boldsymbol{w}$ is also (L', d)-substring distant for every $L < L' \leqslant n$.

Example 1: Let $n = 16$ and let

$$\boldsymbol{x} = 0100000111011111,$$

so its 6-multispectrum is

$$S_6(\boldsymbol{x}) = \{010000, 100000, 000001, 000011, 000111, \\ 001110, 011101, 111011, 110111, 101111, 011111\}.$$

The string $\boldsymbol{x}$ is $(6, 1)$-substring distant, however, $\boldsymbol{x}$ is not $(6, 2)$-substring distant since

$$d_H(\boldsymbol{x}_{7,6}, \boldsymbol{x}_{11,6}) = d_H(011101, 011111) = 1.$$

The family of (L, d)-substring distant strings and more specifically L-substring unique strings is highly related to the set of reconstructible strings, which is defined next. Namely, a string $\boldsymbol{w} \in \Sigma^n$ is called an ***L-reconstructible string*** if it can be uniquely reconstructed from its L-multispectrum. Hence, $\boldsymbol{w}$ is an L-reconstructible string if for every $\boldsymbol{x} \in \Sigma^n$ where $\boldsymbol{x} \neq \boldsymbol{w}$ it holds that $S_L(\boldsymbol{w}) \neq S_L(\boldsymbol{x})$. For positive integers n, d, L, we denote by $\mathcal{Z}_n(L, d)$ the set of all length-n (L, d)-substring distant strings over Σ. For $d = 1$ we simply denote this value by $\mathcal{Z}_n(L)$. The set of L-reconstructible strings is denoted by $\mathcal{R}_n(L)$. We also let $Z_n(L, d) = |\mathcal{Z}_n(L, d)|$, $Z_n(L) = |\mathcal{Z}_n(L)|$, and $R_n(L) = |\mathcal{R}_n(L)|$.

The following connection between substring unique and reconstructible strings was first established by Ukkonen in [32].

Theorem 2: [32] If a string $\boldsymbol{x} \in \Sigma^n$ is $(L-1)$-substring unique then it is L-reconstructible.

According to Theorem 2, it holds that $\mathcal{Z}_n(L-1) \subseteq \mathcal{R}_n(L)$ and in particular $Z_n(L-1) \leqslant R_n(L)$.

The opposite direction of Theorem 2 does not always hold. In fact, in [15], an encoding scheme that uses the property from Theorem 2 is used in order to encode L-reconstructible strings that are almost $(L-1)$-substring unique. Recently, two encoding schemes of reconstructible binary strings that are also $(L-1)$-substring unique were proposed in [12]. The first scheme is applied for a window length of $L = 2\lceil \log(n) \rceil + 2$ with a single bit of redundancy, and the second one works for windows of length $L = \lceil a \log(n) \rceil$ for $1 < a \leqslant 2$ and its asymptotic rate approaches 1. According to the first scheme, one can deduce that $Z_n(2\lceil \log(n) \rceil + 2) \geqslant 2^{n-1}$ and the second one implies that for all $1 < a \leqslant 2$,

$$\lim_{n \to \infty} \frac{\log_2(Z_n(\lceil a \log(n) \rceil))}{n} = 1.$$

This result is also proved directly in [12], by deriving a lower bound on the number of strings in $\mathcal{Z}_n(\lceil a\log(n)\rceil)$.

The motivation to study (L,d)-substring distant strings originates from the observation that in many cases the L-multispectrum cannot be read error-free. This translates to a stronger property, such as the one given by (L,d)-substring distant, that strings need to satisfy in order to guarantee unique reconstruction in the presence of errors.

Definition 3: Let $\boldsymbol{w} \in \Sigma^n$ be a string and $S_L(\boldsymbol{w})$ is its L-multispectrum. A multiset U is called a **t-losses L-multispectrum of $\boldsymbol{w}$** if $U \subseteq S_L(\boldsymbol{w})$ and $|S_L(\boldsymbol{w})|-|U| \leqslant t$. The **$t$-losses L-multispectrum ball of $\boldsymbol{w}$**, denoted by $\mathcal{B}_{L,t}(\boldsymbol{w})$, is defined to be the multiset

$$\mathcal{B}_{L,t}(\boldsymbol{w}) = \{U \mid U \text{ is a } t\text{-losses } L\text{-multispectrum of } \boldsymbol{w}\}.$$

Example 2: Let $n, \boldsymbol{x}$ be from Example 1 and let $L=8$. The L-multispectrum of $\boldsymbol{x}$ is

$$S_L(\boldsymbol{x}) = \{01000001, 10000011, 00000111, 00001110, 00011101, 00111011, 01110111, 11101111, 11011111\}.$$

The multiset

$$U_1 = \{10000011, 00000111, 00001110, 00011101, 11101111, 11011111\},$$

which equals to $S_L(\boldsymbol{x}) \setminus \{\boldsymbol{x}_{1,L}, \boldsymbol{x}_{5,L}, \boldsymbol{x}_{6,L}\}$, is a 3-losses L-multispectrum of $\boldsymbol{x}$. The multiset

$$U_2 = \{01000001, 00001110, 00011101, 11101111, 11011111\},$$

which equals to $S_L(\boldsymbol{x}) \setminus \{\boldsymbol{x}_{2,L}, \boldsymbol{x}_{3,L}\boldsymbol{x}_{5,L}, \boldsymbol{x}_{6,L}\}$, is a 4-losses L-multispectrum of $\boldsymbol{x}$.

Next, we discuss reconstruction from some lossy multispectrum U of a string $\boldsymbol{w}$. Throughout this paper, we assume unless stated otherwise that $\boldsymbol{w}$ is L-reconstructible and in fact L-substring unique (i.e. all substrings of $S_L(\boldsymbol{w})$ are unique) since otherwise the question of reconstructing noisy multispectra is irrelevant.

Moreover, notice that if successive substrings of $\boldsymbol{w}$ are missing from the start or the end of U then several entries of the input string can be entirely absent from U. For example, if the three windows $\boldsymbol{w}_{1,L}, \boldsymbol{w}_{2,L}, \boldsymbol{w}_{n-L+1,L}$ are lost, it will not be possible to restore the values of w_1, w_2, w_n since these entries do not appear at all in U. Therefore, given a multispectrum U, we define its ***maximal-reconstructible substring***, denoted by $\mathbf{W}_1(U)$, to be the largest consecutive substring of $\boldsymbol{w}$ which its L-prefix and L-suffix appear in U (in our example $\mathbf{W}_1(U) = \boldsymbol{w}_{3,n-3}$). Namely, assume $U = \{\boldsymbol{w}_{i_1,L}, \ldots, \boldsymbol{w}_{i_m,L}\}$ where $m \geqslant n-t$ and $i_1, \ldots, i_m \in [n-L+1]$ are distinct indices, then $\mathbf{W}_1(U)$ is the substring $(w_{i_1}, \ldots, w_{i_m+L-1})$. It will follow that in our constructions the string $\mathbf{W}_1(U)$ is uniquely defined and is independent of the original string $\boldsymbol{w}$. Since there are at most t losses, it is ensured that the length of $\mathbf{W}_1(U)$ is at least $n-t$. Notice that if U contains the substrings $\boldsymbol{w}_{1,L}$ and $\boldsymbol{w}_{n-L+1,L}$, it is ensured that $\mathbf{W}_1(U) = \boldsymbol{w}$.

Example 3: Following Example 2, $\mathbf{W}_1(U_1) = \boldsymbol{x}_{2,15}$ and $\mathbf{W}_1(U_2) = \boldsymbol{x}$.

Accordingly, the following definition presents the family of strings that will be studied in Section III.

Definition 4: A string $\boldsymbol{w}$ is called an **(L,t)-reconstructible string** if for any t-losses L-multispectrum $U \in \mathcal{B}_{L,t}(\boldsymbol{w})$, the maximal-reconstructible substring $\mathbf{W}_1(U)$ can be uniquely reconstructed from U.

Remark 1: Note that in the case of $\mathbf{W}_1(U) \neq \boldsymbol{w}$, the reconstruction decoder discovers that it can only reconstruct a part of the string and reconstructs $\mathbf{W}_1(U)$ exactly.

The string reconstruction problem was motivated by the reading mechanism of DNA sequences. In this process, short substrings are read from the long sequence and are then used to reconstruct the sequence. If the length of the read substrings is L, then it is assumed that on each read one of the $n-L+1$ substrings is read with equal probability. The results in the paper can also be used in order to analyze the reduction in the number of required reads while considering the cases of receiving all and almost all of the $n-L+1$ substrings with high probability. This will be explained next.

Assume first that the complete L-multispectrum $S_L(\boldsymbol{x})$ is required to recover the long sequence $\boldsymbol{x}$. Let $C_{\epsilon,0}$ be such that when reading $M = C_{\epsilon,0}n$ substrings of $\boldsymbol{x}$, all $n-L+1$ substrings in $S_L(\boldsymbol{x})$ are read with probability at least $1-\epsilon$. The value of $C_{\epsilon,0}$ can be lower bounded as follows. The probability that a single length-L substring is not read upon M attempts is

$$P = \left(1 - \frac{1}{n-L+1}\right)^M = \left(1 - \frac{1}{n-L+1}\right)^{C_{\epsilon,0}n} \leqslant e^{-C_{\epsilon,0}}.$$

Thus, the probability that upon M reads not all substrings in the L-multispectrum $S_L(\boldsymbol{x})$ are read can be upper bounded by the union bound as

$$P_0 \leqslant (n-L+1)P = (n-L+1)e^{-C_{\epsilon,0}} \leqslant ne^{-C_{\epsilon,0}}.$$

Hence, in order to guarantee success probability of at least $1-\epsilon$, it suffices that $ne^{-C_{\epsilon,0}} \leqslant \epsilon$, i.e.,

$$C_{\epsilon,0} \geqslant \ln(n) + \ln(1/\epsilon).$$

On the other hand, if it is possible to reconstruct the sequence $\boldsymbol{x}$ even in the presence of t losses, then a reconstruction failure occurs when at least $t+1$ substrings are not read. Hence this probability is given by

$$P_t \leqslant \binom{n-L+1}{t+1} \cdot P^{t+1} \leqslant (ne^{-C_{\epsilon,t}})^{t+1},$$

where now the required number of reads is $C_{\epsilon,t}n$. Accordingly, in order to guarantee $1-\epsilon$ success probability, it is enough to require that $(ne^{-C_{\epsilon,t}})^{t+1} \leqslant \epsilon$ and hence

$$C_{\epsilon,t} \geqslant \ln(n) + \frac{\ln(1/\epsilon)}{t+1}.$$

Hence, if for example $1/\epsilon = \mathcal{O}(n^a)$, then the number of reads can be reduced roughly by a factor of $(t+1)(1-\frac{t}{a+t+1})$.

Lastly, for the convenience of the reader, relevant notations and terminology referred to throughout the paper are summarized in Table I.

TABLE I

TABLE OF DEFINITIONS AND NOTATIONS

Notation	Meaning	Remarks
n	String length	Sec. II
Σ	The binary alphabet $\{0,1\}$	Sec. II
$\boldsymbol{w}$	A length-n string	Sec. II
$\mathcal{B}_r(\boldsymbol{w})$	The radius-r ball around $\boldsymbol{w}$	Sec. II
L	Substring length	Sec. II
$S_L(\boldsymbol{w})$	L-multispectrum of $\boldsymbol{w}$	Sec. II
$\mathcal{Z}_n(L,d)$	The set of (L,d)-substring distant length-n strings	Sec. II
$Z_n(L,d)$	The size of $\mathcal{Z}_n(L,d)$	Sec. II
$\mathcal{R}_n(L)$	The set of L-reconstructible length-n strings	Sec. II
$R_n(L,d)$	The size of $\mathcal{R}_n(L)$	Sec. II
$\mathcal{B}_{L,t}(\boldsymbol{w})$	The t-losses L-multispectrum ball of $\boldsymbol{w}$	Def. 3
$\mathbf{W}_1(U)$	The maximal-reconstructible substring of a lossy multispectrum U	Sec. II
ℓ_1	The value $L-\lfloor t/3 \rfloor -1$	Sec. III
ℓ_2	The value $L-\lceil 2t/3 \rceil -1$	Sec. III
ℓ_3	The value $L-t-1$	Sec. III
I_2	The set $[n-\ell_2-t+1, n-\ell_2+1]$	Sec. III
I_3	The set $[n-\ell_3-t+1, n-\ell_3+1]$	Sec. III
$\mathcal{D}_n(L,t)$	The set of length-n strings that satisfy (n,L,t)-LREC constraints	Sec. III
$D_n(L,t)$	The size of $\mathcal{D}_n(L,t)$	Sec. III
$\mathcal{B}_{L,t,s}(\boldsymbol{w})$	The (t,s)-erroneous L-multispectrum ball of $\boldsymbol{w}$	Def. 14
$\mathbf{W}_2(U)$	The majority maximal-reconstructible substring of an erroneous multispectrum U	Sec. IV
$\boldsymbol{u}_d$	A d-auto cyclic string	Sec. IV
$\mathcal{CB}_{n,r}(\boldsymbol{w})$	The concatenation ball of radius-r around $\boldsymbol{w}$	Sec. IV
$\mathcal{D}_n(L,t,s)$	The set of length-n strings that satisfy (n,L,t,s)-EREC constraints	Sec. V
$D_n(L,t,s)$	The size of $\mathcal{D}_n(L,t,s)$	Sec. V
$\mathbf{W}_3(U)$	The consensus maximal-reconstructible substring of an erroneous multispectrum U	Sec. V
$I(V)$	The set $[i_1, i_m+L-1]$ for $V=\{\boldsymbol{u}_{i_1},\ldots,\boldsymbol{u}_{i_m}\}$	Sec. V
$n(V)$	The size of $I(V)$	Sec. V
$SB_k(\boldsymbol{w})$	The k-striping ball of $\boldsymbol{w}$	Sec. V

III. RECONSTRUCTING AN INCOMPLETE MULTISPECTRUM

In this section, we define constraints for (L,t)-reconstructible strings, propose a reconstruction algorithm for those strings and analyze the cardinality of such family of strings.

A. Reconstruction Constraints

The goal of this subsection is to construct t-losses L-reconstructible strings. This will be given by strings that satisfy a few constraints, given in the next definition. For simplicity, we consider here only the binary case, so $\Sigma = \{0,1\}$.

For the rest of this section, we denote the integers $\ell_1 = L-\lfloor t/3\rfloor-1, \ell_2 = L-\lceil 2t/3\rceil-1, \ell_3 = L-t-1$ and the sets $I_2 = [n-\ell_2-t+1, n-\ell_2+1], I_3 = [n-\ell_3-t+1, n-\ell_3+1]$.

Definition 5: A string $\boldsymbol{x} \in \Sigma^n$ is said to satisfy the (n,L,t)-**lossy reconstruction (LREC) constraints** if it fulfills the following three constrains.

1) $\boldsymbol{x}$ is a ℓ_1-substring unique string.
2) The first and last $t+1$ length-ℓ_2 substrings are not identical to all other length-ℓ_2 substrings. Namely, for all $i \in [t{+}1], j \in [n-\ell_2+1]$ with $i \neq j$ then $\boldsymbol{x}_{i,\ell_2} \neq \boldsymbol{x}_{j,\ell_2}$ and for all $i \in [n-\ell_2+1], j \in I_2$ with $i \neq j$, then $\boldsymbol{x}_{i,\ell_2} \neq \boldsymbol{x}_{j,\ell_2}$.
3) The first $t+1$ length-ℓ_3 substrings are not identical to the last $t+1$ length-ℓ_3 substrings. Namely, for all $i \in [t{+}1], j \in I_3$, $\boldsymbol{x}_{i,\ell_3} \neq \boldsymbol{x}_{j,\ell_3}$.

For n, L, t, denote by $\mathcal{D}_n(L,t)$ the set of all strings that satisfy the (n,L,t)-LREC constraints and let $D_n(L,t) = |\mathcal{D}_n(L,t)|$. Note that by definition, if a string satisfies the (n,L,t)-LREC constraints it satisfies the (n,L,t')-LREC constraint for all $t' \leqslant t$, that is, $\mathcal{D}_n(L,t) \subseteq \mathcal{D}_n(L,t')$.

Example 4: Let $n, L, \boldsymbol{x}$ from Example 2. The string $\boldsymbol{x}$ satisfies the $(n,L,4)$-LREC constraints. The first constraint follows from the fact that $\boldsymbol{x}$ is 6-substring unique and it is possible to verify that the two other constraints are satisfied as well. Therefore, $\boldsymbol{x} \in \mathcal{D}_n(L,4)$ and also $\boldsymbol{x} \in \mathcal{D}_n(L,3)$.

In [15], the authors focused on a type of errors which corresponds to occurrence of bursts of substring losses. They identified a lossy multispectrum $U \subseteq S_L(x)$ to have G-maximal coverage gap if G is the maximum number of consecutive substrings that are not included in $S_L(\boldsymbol{x})$. Based on this characterization, they showed that if $\boldsymbol{x}$ is $(L-G-1)$-substring unique it is reconstructible from such a lossy multispectrum U. When applying this constraint to our problem, assume that $U \in \mathcal{B}_{L,t}(\boldsymbol{x})$, then it is necessary that $G = t$ since all the losses can occur consecutively. Hence, the construction proposed in [15] requires that $\boldsymbol{x}$ is $(L-t-1)$-substring unique in order for it to be (L,t)-reconstructible. Based on the results of [12], in order to construct a rate-$(1-o(1))$ code of $(L-t-1)$-substring unique length-n strings, it is required that $L > a\log(n)+t$ for some $a > 1$. It will be shown in Section III-C that the (n,L,t)-LREC constraint composes a rate-$(1-o(1))$ code for values of L that satisfies $L > a\log(n)+t/3$, where $a > \max\{b/3+1, 2b/3\}$ and $t = b\log(n)+o(\log(n))$ for $b \in \mathbb{N}$. Hence, for these parameters, the construction proposed in this paper imposes a weaker constraint on the value of L than the construction proposed in [15].

Moreover, in [8], the authors studied the problem of reconstructing an incomplete multispectrum under the name of *assembly of words in presence of coverage errors*. They presented constructions with an asymptotic rate that is strictly less than 1, that achieve the following results: 1. For $L = c\log(n)$, where $c > 1$, the construction is able to tolerate $(c-2)\log(n)$ losses but the asymptotic rate approaches $(1-2/c)$, and 2. For $L = cn$ with $0 < c < 1$, the construction is able to tolerate $cn - o(n)$ losses with an asymptotic rate that approaches $1-2c$. Thus, for $t = b\log(n) + o(\log(n))$ losses where $b \in \mathbb{N}$, the construction proposed in this paper has higher rate than the one proposed in [8] and requires smaller L.

Algorithm 1 Stitch(A, ρ)

```
for $k = 0, \dots, \rho$ do
    $B = \emptyset$
    while $A \neq \emptyset$ do
        pick $\boldsymbol{w} \in A$ such that for every other $\boldsymbol{w}' \in A$, $\text{Pref}_{L-k-1}(\boldsymbol{w}) \neq \text{Suff}_{L-k-1}(\boldsymbol{w}')$
        set $A = A \setminus \{\boldsymbol{w}\}$
        while there exists $\boldsymbol{w}' \in A$ such that $\text{Suff}_{L-k-1}(\boldsymbol{w}) = \text{Pref}_{L-k-1}(\boldsymbol{w}')$ do
            set $\boldsymbol{w} = \boldsymbol{w} \circ \text{Suff}_{|\boldsymbol{w}'|-L+k+1}(\boldsymbol{w}')$
            set $A = A \setminus \{\boldsymbol{w}'\}$
        end while
        $B = B \cup \{\boldsymbol{w}\}$
    end while
    $A = B$
end for
return $B$
```

Algorithm 2 Reconstruct(U, t)

```
Input: $U \in \mathcal{B}_{L,t}(\boldsymbol{x})$ for some $\boldsymbol{x} \in \mathcal{D}_n(L,t)$
Output: $\mathbf{W}_1(U)$ the maximum reconstructible-substring of $U$
1: Invoke $A_0 = \text{Stitch}(U, \lfloor t/3 \rfloor)$.
2: If $|A_0| = 1$ and $A_0 = \{\boldsymbol{y}\}$: return $\boldsymbol{y}$.
3: If $|A_0| = 2$ and $A_0 = \{\boldsymbol{y}_1, \boldsymbol{y}_2\}$: return Stitch($A_0, t$).
4: If $|A_0| = 3$ and $A_0 = \{\boldsymbol{y}_1, \boldsymbol{y}_2, \boldsymbol{y}_3\}$: for $i = 1,2,3$ invoke
   $A_i = \text{Stitch}(A_0 \setminus \{\boldsymbol{y}_i\}, \lceil 2t/3 \rceil)$ and if successful invoke
   $A_i' = \text{Stitch}(A_i \cup \{\boldsymbol{y}_i\}, \lceil 2t/3 \rceil)$. If successful again, return
   $A_i'$.
```

B. Reconstruction Algorithm

Our next goal is showing that every string which satisfies the (n,L,t)-LREC constraint is an (L,t)-reconstructible string, that is, its maximal-reconstructible substring can be uniquely decoded even if at most some t substrings are not read. Namely, we prove the following theorem.

Theorem 6: Every string $\boldsymbol{x} \in \mathcal{D}_n(L,t)$ is an (L,t)- reconstructible string.

The proof of Theorem 6 is given by an explicit decoding algorithm which receives a multiset $U \in \mathcal{B}_{L,t}(\boldsymbol{x})$ for some $\boldsymbol{x} \in \mathcal{D}_n(L,t)$. First, we present in Algorithm 1, an auxiliary procedure, called the *Stitching Algorithm*, which receives two inputs: 1) A set A of substrings that we aim to stitch, and 2) $\rho \leqslant t$, a parameter that will indicate the minimum overlapping size of two substrings in order to be stitched together. The stitching algorithm is based on iterative stitching steps and is composed of three nested loops. At the most inner loop, two substrings are stitched if the suffix of the first is identical to the prefix of the second. This will later indicate that these substrings originated from the same positions in the input string. The middle loop constructs continuous substrings of U by finding a prefix of such a substring and repeatedly applying the inner loop in order to correctly concatenate to it more bits. The outer loop iterates over $k = 0, \dots, \rho$ and at every iteration we bridge gaps that were created by losses of k consecutive substrings. This is accomplished by reducing the substring length used at the suffix-prefix matching condition method of the inner loop. The stitching algorithm returns a set of continuous substrings reconstructed from U. which its size is smaller than the input set size, or equal if no stitching occurred. We say that an operation of the stitching algorithm is *successful* if the output set size is strictly smaller than the input set size.

Algorithm 2, called the *Reconstruction Algorithm*, receives a t-losses $L-$multispectrum U for some $\boldsymbol{x} \in \mathcal{D}_n(L,t)$ and uses the stitching algorithm to reconstruct $\mathbf{W}_1(U)$, the maximal reconstructible substring of U. In case the returned set by the reconstruction algorithm consists of a single string we assume that the output is the string itself (i.e. not a set).

Assume that $U = \{\boldsymbol{x}_{i_1,L}, \dots, \boldsymbol{x}_{i_m,L}\}$, where $1 \leqslant i_1 < i_2 < \cdots < i_m \leqslant n-L+1$. Let $A_0 = \text{Stitch}(U, \lfloor t/3 \rfloor)$ be the resulting set after Step 1, and denote $A_0 = \{\boldsymbol{y}_1, \dots, \boldsymbol{y}_r\}$. Additionally, denote by B_k the set B after the k-th iteration of the for loop of Algorithm 1. Note that every substring in U is a substring of *exactly* one string from A_0, and that $A_0 = B_{\lfloor t/3 \rfloor}$. The next two examples demonstrate how Algorithms 1 and 2 operate.

Example 5: Let $n, L, \boldsymbol{x}, U_1$ from Example 2, so that $U_1 \in \mathcal{B}_{L,t}(\boldsymbol{x})$, where $t = 3$. Assume that we invoke Reconstruct(U_1, t). First, the algorithm invokes $A_0 = \text{Stitch}(U_1, 1)$. At the first iteration of the for loop where $k = 0$, assume the algorithm picks $\boldsymbol{x}_{2,8} = 10000011$ and stitches to it $\boldsymbol{x}_{3,8} = 00000111$ followed by $\boldsymbol{x}_{4,8} = 00001110$. Next, the algorithm picks $\boldsymbol{x}_{7,8} = 01110111$ and stitches to it $\boldsymbol{x}_{8,8} = 11101111$ followed by $\boldsymbol{x}_{9,8} = 11011111$. Thus, we have at the end of this iteration

$$B_0 = \{\boldsymbol{x}_{2,10}, \boldsymbol{x}_{7,10}\} = \{1000001110, 0111011111\}.$$

No stitching is made at the second iteration for $k = 1$ and thus $A_0 = B_1 = B_0$ is the output of the stitching algorithm. Since $|A_0| = 2$, we execute next in Step 3, Stitch(A_0, 3). Then, the two substrings of A_0 are stitched at iteration $k = 2$, since $\text{Suff}_5(\boldsymbol{x}_{2,10}) = \text{Pref}_5(\boldsymbol{x}_{7,10})$. Eventually, the string

$$\boldsymbol{x}_{2,15} = 1000001110111 = \mathbf{W}_1(U_1)$$

is returned as expected.

Example 6: Let $n, L, \boldsymbol{x}, U_2$ from Example 2, so that $U_2 \in \mathcal{B}_{L,t}(\boldsymbol{x})$, where $t = 4$. Invoking Stitch(U_2, 1) returns

$$\begin{aligned} A_0 = B_1 = B_0 =& \{\boldsymbol{x}_{1,8}, \boldsymbol{x}_{4,8}, \boldsymbol{x}_{7,10}\} \\ =& \{10000011, 00011101, 0111011111\}. \end{aligned}$$

Since $|A_0| = 3$, the reconstruction algorithm executes Step 4. Assume that $\boldsymbol{y}_1 = \boldsymbol{x}_{4,8}, \boldsymbol{y}_2 = \boldsymbol{x}_{1,8}$, and $\boldsymbol{y}_3 = \boldsymbol{x}_{7,10}$. For $i = 1$, the algorithm receives that Stitch$(A_0 \setminus \{\boldsymbol{y}_1\}, 3) = A_0 \setminus \{\boldsymbol{y}_1\}$ which yields with an unsuccessful result. However, for $i = 2$, when invoking Stitch$(A_0 \setminus \{\boldsymbol{y}_2\}, 3)$, the algorithm stitches the substrings $\boldsymbol{y}_1, \boldsymbol{y}_3$ at iteration $k = 2$ using $\text{Suff}_5(\boldsymbol{x}_{1,8}) = \text{Pref}_5(\boldsymbol{x}_{4,8})$, and returns

$$A_2 = \{\boldsymbol{x}_{1,11}\} = \{10000011101\}.$$

Lastly, the algorithms applies Stitch$(A_2 \cup \{\boldsymbol{y}_2\}, 3)$ and stitches again at iteration $k = 2$ to receive $A_2' = \boldsymbol{x} = \mathbf{W}_1(U_2)$ as the final result.

The correctness of Algorithms 1 and 2 is proved in the next few claims.

Claim 7: For all $1 \leqslant j \leqslant m-1$ if $i_{j+1} - i_j \leqslant \lfloor t/3 \rfloor + 1$, then $\boldsymbol{x}_{i_j,L}$ and $\boldsymbol{x}_{i_{j+1},L}$ are substrings of the same string in A_0.

Proof: First, from Constraint 1 it follows that

$$\forall k \in [0, \lfloor t/3 \rfloor], \boldsymbol{x} \text{ is } (L-k-1)\text{-substring unique.} \quad (1)$$

We first claim that for every $k < k' = i_{j+1} - i_j - 1$, $\boldsymbol{x}_{i_j,L}, \boldsymbol{x}_{i_{j+1},L}$ are not substrings of the same string in B_k and furthermore, $\boldsymbol{x}_{i_j,L}$ is a suffix of some substring $\boldsymbol{w}_1 \in B_k$, while $\boldsymbol{x}_{i_{j+1},L}$ is a prefix of another substring $\boldsymbol{w}_2 \in B_k$. To see this, assume in the contrary and let $k_0 < k'$ be the first iteration where $\boldsymbol{x}_{i_j,L}$ is not a suffix of some substring in B_{k_0}. That is, there exists $\boldsymbol{w} \in B_{k_0-1}$ where $\text{Suff}_L(\boldsymbol{w}) = \boldsymbol{x}_{i_j,L}$, that is stitched to the left of another $\boldsymbol{w}' \in B_{k_0-1}$, which satisfies $\text{Pref}_L(\boldsymbol{w}') = \boldsymbol{x}_{i_g,L}$ for another substring $\boldsymbol{x}_{i_g,L} \in U$. However, it follows from (1) that $i_j < i_g < i_{j+1}$ and therefore such a substring cannot exist in U.

In particular, $\boldsymbol{x}_{i_j,L}, \boldsymbol{x}_{i_{j+1},L}$ are not substrings of the same string in B_k. Thus, at the k'-th iteration, the substrings are stitched since

$$\begin{aligned}\text{Suff}_{L-k'-1}(\boldsymbol{w}_1) &= \boldsymbol{x}_{i_j+k'+1,L-k'-1} = \boldsymbol{x}_{i_{j+1},L-k'-1} \\ &= \text{Pref}_{L-k'-1}(\boldsymbol{w}_2).\end{aligned}$$

Note that from (1), for any other substring $\boldsymbol{w}_3 \notin \{\boldsymbol{w}_1, \boldsymbol{w}_2\}$, $\text{Suff}_{L-k'-1}(\boldsymbol{w}_1) \neq \text{Pref}_{L-k'-1}(\boldsymbol{w}_3)$ and $\text{Pref}_{L-k'-1}(\boldsymbol{w}_2) \neq \text{Suff}_{L-k'-1}(\boldsymbol{w}_3)$. ■

It is said that a spectrum U experienced a *burst of losses* of length h at index $i \leqslant n-L-t+1$ if $\boldsymbol{x}_{i,L}, \ldots, \boldsymbol{x}_{i+h-1,L} \notin U$.

Claim 8: The set $A_0 = \text{Stitch}(U, \lfloor t/3 \rfloor)$ satisfies $|A_0| \leqslant 3$.

Proof: Following Claim 7, we will show that there are three possible cases for the size of the set A_0. First, if there are no bursts of losses longer than $\lfloor t/3 \rfloor$, then all substrings of U are contained in a single string of A_0, thus $|A_0| = 1$. Second, if there is a single burst of losses longer than $\lfloor t/3 \rfloor$, then the substrings of U are divided into two different strings of A_0, thus $|A_0| = 2$. Similarly, at the third case there are two bursts of losses longer than $\lfloor t/3 \rfloor$, and then $|A_0| = 3$. Other cases are not possible, since the number of losses is at most t. ■

Claim 9: At Step 3 of Algorithm 2, the result of Stitch(A_0, t) is $\mathbf{W}_1(U)$.

Proof: Let $A_0 = \{\boldsymbol{y}_1, \boldsymbol{y}_2\}$, where $\text{Suff}_L(\boldsymbol{y}_1) = \boldsymbol{x}_{i_j,L}$ and $\text{Pref}_L(\boldsymbol{y}_2) = \boldsymbol{x}_{i_{j+1},L}$. It follows that $\text{Pref}_L(\boldsymbol{y}_1) = \boldsymbol{x}_{i_1,L}$, where $i_1 \leqslant t+1$. Therefore, the prefix $\text{Pref}_{L-t-1}(\boldsymbol{y}_1)$ is one of the first $t+1$ length-$(L-t-1)$ substrings of $\boldsymbol{x}$. Similarly, $\text{Suff}_L(\boldsymbol{y}_2) = \boldsymbol{x}_{i_m,L}$, where $i_m \geqslant n-L-t+1$, and the suffix $\text{Suff}_{L-t-1}(\boldsymbol{y}_2)$ is one of the last $t+1$ length-$(L-t-1)$ substrings of $\boldsymbol{x}$. Thus, from Constraint 3, for every $k \in [0,t]$, $\text{Pref}_{L-k-1}(\boldsymbol{y}_1) \neq \text{Suff}_{L-k-1}(\boldsymbol{y}_2)$. Therefore, it is not possible to stitch the substring $\boldsymbol{y}_1$ to the right of $\boldsymbol{y}_2$. Since there are at most t losses, it follows that $i_{j+1} - i_j - 1 \leqslant t$. Hence, these substrings are stitched correctly to a single string at iteration $k' = i_{j+1} - i_j - 1$, which results with the string $\mathbf{W}_1(U)$. ■

Claim 10: At Step 4 of Algorithm 2, there exists a substring $\boldsymbol{y}_i \in A_0$ such that both operations of the stitching algorithm are successful. For such a $\boldsymbol{y}_i$, the result of this step is the string $\mathbf{W}_1(U)$.

Proof: Let $A_0 = \{\boldsymbol{y}_1, \boldsymbol{y}_2, \boldsymbol{y}_3\}$ where $\text{Suff}_L(\boldsymbol{y}_1) = \boldsymbol{x}_{i_j,L}$, $\text{Pref}_L(\boldsymbol{y}_2) = \boldsymbol{x}_{i_{j+1},L}$, $\text{Suff}_L(\boldsymbol{y}_2) = \boldsymbol{x}_{i_h,L}$, and $\text{Pref}_L(\boldsymbol{y}_3) = \boldsymbol{x}_{i_{h+1},L}$. Since the number of losses is at most t, it follows that $\lfloor t/3 \rfloor < i_{j+1} - i_j - 1 \leqslant \lceil 2t/3 \rceil$ and $\lfloor t/3 \rfloor < i_{h+1} - i_h - 1 \leqslant \lceil 2t/3 \rceil$. Similarly to the proof of Claim 9, but in this case according to Constraint 2, for every $k \in [0, \lceil 2t/3 \rceil]$,

$$\text{Pref}_{L-k-1}(\boldsymbol{y}_1) \neq \ \text{Suff}_{L-k-1}(\boldsymbol{y}_s) \text{ for } s \in [2,3], \quad (2)$$

$$\text{Suff}_{L-k-1}(\boldsymbol{y}_3) \neq \ \text{Pref}_{L-k-1}(\boldsymbol{y}_s) \text{ for } s \in [1,2]. \quad (3)$$

Thus, if we pick $\boldsymbol{y}_1$, from (3) we can only stitch $\boldsymbol{y}_2$ to the left of $\boldsymbol{y}_3$ at Stitch$(A_0 \setminus \{\boldsymbol{y}_1\}, \lceil 2t/3 \rceil)$ and from (2) we stitch the result to the right of $\boldsymbol{y}_1$ at Stitch$(A_1 \cup \{\boldsymbol{y}_1\}, \lceil 2t/3 \rceil)$. The result is similar if we initially pick $\boldsymbol{y}_3$. If we pick $\boldsymbol{y}_2$ it follows that it is only possible to incorrectly stitch $\boldsymbol{y}_1$ to the left of $\boldsymbol{y}_3$ at Stitch$(A_0 \setminus \{\boldsymbol{y}\}, \lceil 2t/3 \rceil)$. However, it is ensured from (2) and (3) that the resulting string in this case cannot be stitched to $\boldsymbol{y}_2$ at the second operation of the stitching algorithm.

Since both operations are successful, the result contains a single substring which contains all the substrings of U. ■

Lemma 11: Algorithm 2 uniquely reconstructs $\mathbf{W}_1(U)$.

Proof: Following Claim 8, there are three possible cases for the size of A_0. From Claims 7, 9, and 10, Algorithm 2 returns a single string where all the elements of U are a substrings of it. That is the maximum reconstructible-substring of U. ■

Lemma 11 verifies also the proof of Theorem 6.

C. Cardinality Analysis

Our next goal is to estimate the value of $D_n(L,t)$ for some specific parameters of n, L, t. Our approach is based on the probabilistic method used in [13] to prove that the asymptotic rate of the set $\mathcal{Z}_{n,2}(L)$ approaches 1, when $L = \lceil a \log(n) \rceil$ and $a > 1$. Building upon this approach, for a given value of t that satisfies $t = b\log(n) + o(\log(n))$ for some $b \in \mathbb{N}$, we show how to choose the value of L such that the three (n,L,t)-LREC constraints hold. This result is proved in the following theorem.

Theorem 12: Let $t = b\log(n) + o(\log(n))$ for $b \in \mathbb{N}$ and $L = \lceil a\log(n) \rceil + \lfloor t/3 \rfloor + 1$ where $a > \max\{b/3 + 1, 2b/3\}$, then it holds that

$$\lim_{n \to \infty} \frac{\log(D_n(L,t))}{n} = 1.$$

We follow a similar outline as the proof presented in [13] for L-substring unique strings, and prove Theorem 12 using the asymmetric Loàsz local lemma which was first proved in [14] and is stated next as it appears in [3].

Lemma13 ([3], Lemma 5.1.1): Let $Y_0, \ldots, Y_{m-1}$ be events in the arbitrary probability space. Let $G = (V, E)$ be a graph with $V = [m]$ such that for every $i \in [m]$, the event Y_i is mutually independent of all the events $\{Y_j \mid (i,j) \notin E\}$. Suppose that there are real numbers $\alpha_0, \ldots, \alpha_{m-1}$ such that $\alpha_i \in [0,1]$ and for all $i \in [m]$,

$$Pr(Y_i) \leqslant \alpha_i \prod_{(i,j)\in E} (1-\alpha_j).$$

Then, it is satisfied that

$$Pr\left(\bigcap_{i\in[m]} \overline{Y}_i\right) \geqslant \prod_{i\in[m]} (1-\alpha_i)$$

where $\overline{Y}_i$ is the complement of Y_i.

Proof of Theorem 12: For the values of t and L stated in the theorem it holds that $L = (a + b/3)\log(n) + o(\log(n))$, $\ell_1 = a\log(n) + o(\log(n))$, $\ell_2 = (a - b/3)\log(n) + o(\log(n))$, and $\ell_3 = (a - 2b/3)\log(n) + o(\log(n))$. Note that for simplicity, we assume all are integer values. The size of $\mathcal{D}_n(L,t)$ will be estimated by a probabilistic approach. Assume that $\boldsymbol{w}$ is a length-n string chosen uniformly at random over Σ. In order to estimate the value of $D_n(L,t)$, we may estimate the probability $Pr(\boldsymbol{w} \in \mathcal{D}_n(L,t))$ since $D_n(L,t) = 2^n \cdot Pr(\boldsymbol{w} \in \mathcal{D}_n(L,t))$ and hence

$$\lim_{n\to\infty} \frac{\log(D_n(L,t))}{n} = 1 + \lim_{n\to\infty} \frac{1}{n}\log(Pr(\boldsymbol{w} \in \mathcal{D}_n(L,t))). \tag{4}$$

For $\ell \in \{\ell_1, \ell_2, \ell_3\}$ and positions $i, j \in [n-\ell+1]$ we notate $\boldsymbol{z} = (i,j,\ell)$ and denote by $I_{\boldsymbol{z}} = \mathbb{1}(\boldsymbol{w}_{i,\ell} = \boldsymbol{w}_{j,\ell})$ the indicator function of the event that the ℓ-substrings that start at positions i and j are identical. According to the LREC constraints, we denote

$$\Gamma_1 = \{(i,j,\ell_1) \mid i,j \in [n-\ell_1+1] \text{ and } i<j\},$$
$$\Gamma_2 = \left\{(i,j,\ell_2) \left| \begin{array}{l} i \in [t+1], j \in [n-\ell_2+1], i<j \text{ or} \\ i \in [n-\ell_2+1], j \in I_2, i<j \end{array} \right.\right\},$$
$$\Gamma_3 = \{(i,j,\ell_3) \mid i \in [t+1], j \in I_3\},$$

and let $\Gamma = \Gamma_1 \cup \Gamma_2 \cup \Gamma_3$ denote the set of all admissible triples. Note that $|\Gamma_1| \leqslant n^2, |\Gamma_2| \leqslant 2(t+1)n$, and $|\Gamma_3| \leqslant (t+1)^2$. We are interested in a lower bound on

$$Pr(\boldsymbol{w} \in \mathcal{D}_n(L,t)) = Pr\left(\sum_{\boldsymbol{z}\in\Gamma} I_{\boldsymbol{z}} = 0\right).$$

Note that for every $\boldsymbol{z} = (i,j,\ell) \in \Gamma$ it holds that $Pr(I_{\boldsymbol{z}}) = \frac{1}{2^\ell}$. Let $\boldsymbol{z} = (i,j,\ell), \boldsymbol{z}' = (i',j',\ell') \in \Gamma$. It is clear that if the substrings $\boldsymbol{w}_{i,\ell}, \boldsymbol{w}_{j,\ell}$ do not overlap with $\boldsymbol{w}_{i',\ell'}$ or $\boldsymbol{w}_{j',\ell'}$, then the indicators $I_{\boldsymbol{z}}, I_{\boldsymbol{z}'}$ are independent. We use Lemma 13 with a graph $G = (V,E)$ such that $V = \Gamma$ and there is an edge $\boldsymbol{z} \to \boldsymbol{z}'$ if at least one of $\boldsymbol{w}_{i,\ell}, \boldsymbol{w}_{j,\ell}$ overlaps with $\boldsymbol{w}_{i',\ell'}$ or $\boldsymbol{w}_{j',\ell'}$. It follows that every $\boldsymbol{z} = (i,j,\ell) \in \Gamma$ has at most $2(\ell_1 + \ell)n$ neighbors from Γ_1 in the graph.

We set the numbers $\alpha_{\boldsymbol{z}} = \frac{1}{4\ell_1 n}$ for every $\boldsymbol{z} \in \Gamma_1$, $\alpha_{\boldsymbol{z}} = \frac{1}{2(t+1)n}$ for every $\boldsymbol{z} \in \Gamma_2$ and $\alpha_{\boldsymbol{z}} = \frac{1}{(t+1)^2}$ for $\boldsymbol{z} \in \Gamma_3$. For every $\boldsymbol{z} \in \Gamma$ it holds that

$$\begin{aligned}
&\prod_{(\boldsymbol{z},\boldsymbol{z}')\in E} (1-\alpha_{\boldsymbol{z}'}) \\
&\geqslant \prod_{(\boldsymbol{z},\boldsymbol{z}')\in E:\boldsymbol{z}'\in\Gamma_1} (1-\alpha_{\boldsymbol{z}'}) \prod_{\boldsymbol{z}'\in\Gamma_2} (1-\alpha_{\boldsymbol{z}'}) \prod_{\boldsymbol{z}'\in\Gamma_3} (1-\alpha_{\boldsymbol{z}'}) \\
&\geqslant \left(1-\frac{1}{4\ell_1 n}\right)^{4\ell_1 n} \left(1-\frac{1}{2(t+1)n}\right)^{2(t+1)n} \left(1-\frac{1}{(t+1)^2}\right)^{(t+1)^2} \\
&\geqslant \frac{1}{e^3},
\end{aligned} \tag{5}$$

since each of the expressions in the last inequality approaches e^{-1} from above as $n \to \infty$. The condition of the lemma holds for every $\boldsymbol{z} \in \Gamma$ since

$$\begin{aligned}
\forall_{\boldsymbol{z}\in\Gamma_1} : Pr(I_{\boldsymbol{z}}) &= \frac{1}{2^{\ell_1}} = \frac{1}{n^a} \overset{\text{(i)}}{\leqslant} \frac{1}{4\ell_1 n} \cdot \frac{1}{e^3} \\
&\leqslant \alpha_{\boldsymbol{z}} \prod_{(\boldsymbol{z},\boldsymbol{z}')\in E} (1-\alpha_{\boldsymbol{z}'}), \\
\forall_{\boldsymbol{z}\in\Gamma_2} : Pr(I_{\boldsymbol{z}}) &= \frac{1}{2^{\ell_2}} = \frac{1}{n^{a-b/3}} \overset{\text{(ii)}}{\leqslant} \frac{1}{2(t+1)n} \cdot \frac{1}{e^3} \\
&\leqslant \alpha_{\boldsymbol{z}} \prod_{(\boldsymbol{z},\boldsymbol{z}')\in E} (1-\alpha_{\boldsymbol{z}'}), \\
\forall_{\boldsymbol{z}\in\Gamma_3} : Pr(I_{\boldsymbol{z}}) &= \frac{1}{2^{\ell_3}} = \frac{1}{n^{a-2b/3}} \overset{\text{(iii)}}{\leqslant} \frac{1}{(t+1)^2} \cdot \frac{1}{e^3} \\
&\leqslant \alpha_{\boldsymbol{z}} \prod_{(\boldsymbol{z},\boldsymbol{z}')\in E} (1-\alpha_{\boldsymbol{z}'}),
\end{aligned}$$

where (i) follows from $4\ell_1\ n = o(n^a)$ since $a > 1$, (i) follows from $2(t+1)n = o(n^{a-b/3})$ since $a - b/3 > 1$, $t = b\log(n) + o(\log(n))$ and (iii) follows from $(t+1)^2 = o(n^{a-2b/3})$ since $t = b\log(n) + o(\log(n))$ and $a - 2b/3 > 0$.

By applying Lemma 13 and from equation (5) we obtain

$$\begin{aligned}
Pr(\boldsymbol{w} \in \mathcal{D}_n(L,t)) &\geqslant \prod_{\boldsymbol{z}\in\Gamma} (1-\alpha_{\boldsymbol{z}}) \\
&\geqslant \frac{1}{e^2} \prod_{\boldsymbol{z}\in\Gamma_1} (1-\alpha_{\boldsymbol{z}}) \\
&\geqslant \frac{1}{e^2} \left(1-\frac{1}{4\ell_1 n}\right)^{n^2}.
\end{aligned}$$

Finally, since

$$\left(1-\frac{1}{4\ell_1 n}\right)^{n^2} \approx \exp\left(-\frac{n}{4a\log(n)}\right),$$

it follows that $\frac{1}{n}\log(Pr(\boldsymbol{w} \in \mathcal{D}_n(L,t)))$ approaches 0 as $n \to \infty$. By plugging into (4) the theorem statement holds. ■

IV. Reconstructing an Erroneous Multispectrum

In this section, we address the problem of reconstructing strings from a multispectrum that suffered substitution errors. This family of multispectra is formally defined as follows.

Definition 14: Let $\boldsymbol{w} \in \Sigma^n$ be a string and $S_L(\boldsymbol{w})$ is its L-multispectrum. A multiset $U = \{\boldsymbol{u}_1, \ldots, \boldsymbol{u}_{n-L+1}\}$ is called a (t,s)**-erroneous** L**-multispectrum of** $\boldsymbol{w}$ if there exists a set of indices $I_e(U)=\{i_1,\ldots,i_m\}\subset[n-L+1]$ where $m\leqslant t$ such that for every $i\in[n-L+1]\setminus I_e(U)$, $\boldsymbol{u}_i=\boldsymbol{w}_{i,L}$ and for every $i\in I_e(U)$, $d_H(\boldsymbol{u}_i,\boldsymbol{w}_{i,L})\leqslant s$. The (t,s)**-erroneous** L**-multispectrum ball of** $\boldsymbol{w}$, denoted by $\mathcal{B}_{L,t,s}(\boldsymbol{w})$, is defined to be the multiset

$$\mathcal{B}_{L,t,s}(\boldsymbol{w})=\{U|U \text{ is a}\,(t,s)\text{-erroneous } L\text{-multispectrum of } \boldsymbol{w}\}.$$

We refer to the substrings of U at positions $I_e(U)$ as the *erroneous substrings* of U, and to the substrings at positions $[n-L+1]\setminus I_e(U)$ as the *correct substrings* of U.

Example 7: Let $n=16, L=10$ and

$$\boldsymbol{x}=1011100010110111.$$

The L-multispectrum of $\boldsymbol{x}$ is

$$S_L(\boldsymbol{x})=\{1011100010, 0111000101, 1110001011, 1100010110, 1000101101, 0001011011, 0010110111\}$$

The multiset

$$U=\{0011100010, 0111000101, 1100001011, 1100011110, 1000101101, 0001011011, 0010110111\}$$

is a $(3,1)$-erroneous L-multispectrum of $\boldsymbol{x}$. That is since by notating the substrings of U as $\boldsymbol{u}_1,\ldots,\boldsymbol{u}_7$, Definition 14 is satisfied for $I_e=\{1,3,4\}$.

Let $\boldsymbol{w}\in\Sigma^n$ be a string and $U=\{\boldsymbol{u}_1,\ldots,\boldsymbol{u}_{n-L+1}\}\in\mathcal{B}_{L,t,s}(\boldsymbol{w})$ be an erroneous spectrum. Note that if an entry of the input string $\boldsymbol{w}$ can appear in U incorrectly more times than it appears correctly, we are not able to determine its correct value from U. Hence, let $\mathbf{W}_2(U)$ denote the maximum reconstructible-substring of U, a string of length n that takes at every position i the majority value of the occurrences of w_i in U. Namely, for a multiset A over Σ we define the function maj(A) that returns the element $a\in A$ with the most appearances in A. If there is more than one element of A that satisfies this requirement, the function maj selects the first element in lexicographic order. For convenience, we sometimes apply the function maj on a vector instead of a multiset. Thus,

$$\mathbf{W}_2(U)=(w'_1,\ldots,w'_n) \text{ with } w'_j=\text{val}(U,j),$$

where

$$\text{val}(U,j)=\text{maj}\{(\boldsymbol{u}_i)_k \mid i\in[n-L+1], k\in[L], i+k-1=j\}.$$

Example 8: Following Example 7,

$$\mathbf{W}_2(U)=0011100010110111,$$

which satisfies $d_H(\mathbf{W}_2(U),\boldsymbol{x})=1$.

Definition 15: A string $\boldsymbol{w}$ is called an (L,t,s)***-reconstructible string*** if for any (t,s)-erroneous L-multispectrum $U\in\mathcal{B}_{L,t,s}(\boldsymbol{w})$, the maximal-reconstructible substring $\mathbf{W}_2(U)$ can be uniquely reconstructed from U.

In order to have a controlled number of incorrect entries in $\mathbf{W}_2(U)$, we add for the rest of this section the constraint $t<L/2$. This constraint ensures that for every $U\in\mathcal{B}_{L,t,s}(\boldsymbol{w})$, all entries of $\boldsymbol{w}$ besides the first and last $2t$ entries cannot appear in U erroneously more times than their appear correctly. Therefore, $\mathbf{W}_2(U)$ satisfies $\mathbf{W}_2(U)_{2t+1,n-4t}=\boldsymbol{w}_{2t+1,n-4t}$. Moreover, it follows that if for every $j\in[2t]\cup[n-2t+1,n]$ the entry w_j equals val(U,j), then $\mathbf{W}_2(U)=\boldsymbol{w}$.

Remark 2: Note that by altering the definitions of multispectrum and (L,d)-substring distant strings to work with cyclic strings and cyclic multispectrum in which the substrings are read cyclically, the issues with the errors in the first and last $2t$ entries of the reconstructed string would have been solved. This will similarly resolve the same problems of incomplete multispectrum in Section III. However, we choose to use the acyclic definitions, since they model better the DNA sequencing problem and are better consistent with previous works such as [12], [15].

In the next subsection we present a reconstruction algorithm for erroneous multispectra of (L,t,s)-reconstructible strings, which is based upon the substring-distant property. Recall that a string $\boldsymbol{w}\in\Sigma^n$ is (L,d)-substring distant if the Hamming distance of its L-multispectrum is at least d. A natural question to ask is whether such strings exist and if so how many as a function of n, L, and d. We address this question in Section IV-B to provide conditions in which the redundancy of this set is at most one bit and when its asymptotic rate approaches 1. Then, in Section IV-C, we present encoding and decoding schemes for such strings which use a single bit of redundancy.

A. Reconstruction Algorithm

The main result of this subsection is summarized in the following theorem.

Theorem 16: If a string $\boldsymbol{x}\in\Sigma^n$ is $(L-1,4s+1)$-substring distant, then it is (L,t,s)-reconstructible for every $t\in[n]$.

Remark 3: Notice that the constraint presented in Theorem 16 is independent of t, as the theorem holds for every $t\in[n]$. This follows from the requirement of $(4s+1)$-distance between the substrings, as seen in the upcoming proof. Alternative constraints, which are not independent of t and require smaller distance between the substrings, are presented in Section V.

The proof of Theorem 16 is given by an explicit reconstruction algorithm, presented in Algorithm 3. The algorithm receives an erroneous multispectrum $U\in\mathcal{B}_{L,t,s}(\boldsymbol{x})$ for $\boldsymbol{x}\in\mathcal{Z}_n(L-1,4s+1)$ and reconstructs the maximum reconstructible substring $\mathbf{W}_2(U)$. The algorithm uses the substring-distant property of $\boldsymbol{x}$ to identify the correct order of the substrings of U. Then, it takes for each entry of $\boldsymbol{x}$ the majority vote of its occurrences in U.

Let $U=\{\boldsymbol{u}_1,\ldots,\boldsymbol{u}_{n-L+1}\}$ be the input set of the algorithm, ordered with respect to $S_L(\boldsymbol{x})$, similarly to Definition 14. A demonstration of the execution of Algorithm 3 is presented in the next example.

Example 9: Let $n,L,\boldsymbol{x},U$ from Example 7. The string $\boldsymbol{x}$ is $(L-1,5)$-substring distant and therefore U is a valid input for Algorithm 3 with $t=3, s=1$. Let $\boldsymbol{u}_1,\ldots,\boldsymbol{u}_7$ denote the elements of U similarly to Example 7. The algorithm picks at Step 2 the substring

$$\boldsymbol{w}_1=\boldsymbol{u}_1=0011100010$$

Algorithm 3 Reconstruct(U, t, s)

```
Input: $U \in \mathcal{B}_{L,t,s}(\boldsymbol{x})$ for $\boldsymbol{x} \in \mathcal{Z}_n(L-1, 4s+1)$
Output: $\mathbf{W}_2(U)$ the maximum reconstructible-substring of $U$
1: Initialize $B[1,\ldots,n]$ as an array of $n$ empty vectors, set $i = 1, A = U$
2: Pick $\boldsymbol{w}_1 \in A$ such that for every other $\boldsymbol{w} \in A$, $d_H(\text{Pref}_{L-1}(\boldsymbol{w}_1),\text{Suff}_{L-1}(\boldsymbol{w})) \geqslant 2s+1$
3: Set $A = A \setminus \{\boldsymbol{w}_1\}$
4: For every $j = 1,\ldots,L$, append $(\boldsymbol{w}_1)_j$ to $B[j]$
5: while $|A| \neq 0$ do
6:     Pick $\boldsymbol{w}_{i+1} \in A$ such that $d_H(\text{Suff}_{L-1}(\boldsymbol{w}_i), \text{Pref}_{L-1}(\boldsymbol{w}_{i+1})) \leqslant 2s$
7:     Set $A = A \setminus \{\boldsymbol{w}_{i+1}\}, i = i + 1$
8:     For every $j = 1,\ldots,L$, append $(\boldsymbol{w}_i)_j$ to $B[i+j-1]$
9: end while
10: Return $\boldsymbol{y} = (y_1,\ldots,y_n)$ where $y_j = maj(B[j])$
```

since for every other $i \in [2, 7]$,

$$d_H(001110001, \text{Suff}_9(\boldsymbol{u}_i)) \geqslant 3.$$

Then, it continues to pick the other substrings of U in increasing order at Step 6, since for every substring $\boldsymbol{u}_i$ for $i \in [1, 6]$, only $\boldsymbol{u}_{i+1}$ satisfies

$$d_H(\text{Suff}_9(\boldsymbol{u}_i), \text{Pref}_9(\boldsymbol{u}_{i+1})) \leqslant 2.$$

For example, both $\boldsymbol{u}_3, \boldsymbol{u}_4$ are erroneous but yet satisfy

$$d_H(\text{Suff}_9(\boldsymbol{u}_3), \text{Pref}_9(\boldsymbol{u}_4)) = d_H(100001011, 110001111) = 2$$

while for every $i \neq 4$,

$$d_H(\text{Suff}_9(\boldsymbol{u}_3), \text{Pref}_9(\boldsymbol{u}_i)) \geqslant 3.$$

Therefore, Algorithm 3 holds for every $j \in [n]$ all the occurrences of x_j in U inside the vector $B[j]$. For example, $B[1] = (0), B[5] = (1, 1, 0, 1, 1)$ and so on. Thus, following the construction of the result string in Step 10, the algorithm returns

$$\boldsymbol{y} = \mathbf{W}_2(U) = 0011100010110111.$$

We prove next the correctness of Algorithm 3.

Lemma 17: Given any $U \in \mathcal{B}_{L,t,s}(\boldsymbol{x})$ for $\boldsymbol{x} \in \mathcal{Z}_n(L - 1, 4s + 1)$, Algorithm 3 successfully returns $\boldsymbol{y} = \mathbf{W}_2(U)$ at Step 10.

Proof: First, we show that the algorithm matches two substrings $\boldsymbol{w}_i, \boldsymbol{w}_{i+1}$ in Step 6 if and only if $\boldsymbol{w}_i = \boldsymbol{u}_j$ and $\boldsymbol{w}_{i+1} = \boldsymbol{u}_{j+1}$ for some consecutive $\boldsymbol{u}_j, \boldsymbol{u}_{j+1} \in U$. That is, since from $U \in \mathcal{B}_{L,t,s}(\boldsymbol{x})$ we have that

$$\begin{aligned} &d_H(\text{Suff}_{L-1}(\boldsymbol{u}_j), \text{Pref}_{L-1}(\boldsymbol{u}_{j+1})) \\ &\quad\leqslant d_H(\text{Suff}_{L-1}(\boldsymbol{u}_j), \boldsymbol{x}_{j+1,L-1}) \\ &\quad\quad + d_H(\text{Pref}_{L-1}(\boldsymbol{u}_{j+1}), \boldsymbol{x}_{j+1,L-1}) \\ &\quad\leqslant s + s = 2s, \end{aligned}$$

and on the other hand, for $\boldsymbol{u}_j, \boldsymbol{u}_k \in U$ with $k \neq j + 1$, it follows from $\boldsymbol{x} \in \mathcal{Z}_n(L-1, 4s+1)$ and from $U \in \mathcal{B}_{L,t,s}(\boldsymbol{x})$ that

$$\begin{aligned} &d_H(\text{Suff}_{L-1}(\boldsymbol{u}_j), \text{Pref}_{L-1}(\boldsymbol{u}_k)) \\ &\quad\geqslant d_H(\boldsymbol{x}_{j+1,L-1}, \boldsymbol{x}_{k,L-1}) - d_H(\text{Suff}_{L-1}(\boldsymbol{u}_j), \boldsymbol{x}_{j+1,L-1}) \\ &\quad\quad - d_H(\text{Pref}_{L-1}(\boldsymbol{u}_k), \boldsymbol{x}_{k,L-1}) \\ &\quad\geqslant 4s + 1 - s - s = 2s + 1. \end{aligned}$$

Using the same arguments, we pick at Step 2 $\boldsymbol{w}_1 = \boldsymbol{u}_1$. Using a simple induction, it follows that for every $i \in [n - L + 1]$, $\boldsymbol{w}_i = \boldsymbol{u}_i$. Hence, for every $j \in [n]$,

$$B[j] = \{(\boldsymbol{u}_i)_k \mid i \in [n - L + 1], k \in [L], i\ + k - 1 = j\},$$

and therefore the string constructed by the algorithm in Step 10 is $\mathbf{W}_2(U)$. ■

The proof of Lemma 17 also verifies the correctness of Theorem 16.

B. Cardinality Analysis of (L, d)-Substring Distant Strings

In this subsection we study the cardinality of the set of substring distant strings for different parameters of L and d. For simplicity, all the results in the of this section are presented for the binary case. The next lemma assures that for given d and for n large enough, the redundancy of the set $\mathcal{Z}_n(L, d)$ is at most a single bit, when $L = 2\log(n)+(d-1)\log(\log(n))+\mathcal{O}(1)$.

Lemma 18: For fixed d, $L = 2\log(n)+(d-1)\log(\log(n))+\mathcal{O}(1)$ and n large enough, it holds that $Z_n(L, d) \geqslant 2^{n-1}$ and hence the redundancy of the set $\mathcal{Z}_n(L, d)$ is at most a single bit.

Proof: Let $L = 2\log(n) + (d-1)\log(\log(n)) + C(d-1)$ for some positive constant C that will be determined later. If a string is not an (L, d)-substring distant, then it contains at least two length-L substrings which their Hamming distance is at most $d - 1$. Notice that the size of the radius-$(d - 1)$ Hamming ball around any $\boldsymbol{v} \in \Sigma^L$ is

$$\begin{aligned} |\mathcal{B}_{d-1}(\boldsymbol{v})| &= \sum_{i=0}^{d-1} \binom{L}{i} \leqslant d \cdot \frac{L(L-1)\cdots(L-d+2)}{(d-1)!} \\ &\leqslant \frac{L^{d-1}}{(d-1)!/d} \leqslant L^{d-1} + 1. \end{aligned}$$

Hence, according to the union bound, the number of strings that are not (L, d)-substring distant can be bounded above by

$$\begin{aligned} n^2 2^{n-L} L^{d-1} &= 2^n \frac{n^2(L^{d-1}+1)}{2^L} = 2^n \frac{L^{d-1}+1}{(\log(n))^{d-1} 2^{C(d-1)}} \\ &\overset{(a)}{\leqslant} 2^n \left(\frac{3\log(n)}{2^C \log(n)}\right)^{d-1} = 2^n \left(\frac{3}{2^C}\right)^{d-1}, \end{aligned}$$

where inequality (a) holds for n large enough. Hence, by choosing $C = \log(3) + 1/(d - 1)$ we get that the number of strings that are not (L, d)-substring distant is at most 2^{n-1}, which accordingly implies that $Z_n(L, d) \geqslant 2^{n-1}$. ■

Our next result claims that the asymptotic rate of the set $\mathcal{Z}_n(L, d)$, when $L = \lceil a\log(n)\rceil$ and $a > 1$, is 1. The proof follows the same outline as the proof presented in [13] for L-substring unique strings, and as the proof of Theorem 12, and uses the asymmetric Loàsz local lemma as stated in Lemma 13. We present the proof of this theorem here as well for the completeness of the results in the paper.

Theorem 19: For fixed d, $a > 1$, and $L = \lceil a\log(n)\rceil$, it holds that the asymptotic rate of the set $\mathcal{Z}_n(L,d)$ is 1. That is,

$$\lim_{n\to\infty} \frac{\log(Z_n(L,d))}{n} = 1.$$

Proof: The size of $\mathcal{Z}_n(L,d)$ will be estimated by a probabilistic approach. Assume that $\boldsymbol{w}$ is a length-n string chosen uniformly at random over Σ. In order to estimate the size $Z_n(L,d)$, we may estimate the probability $Pr(\boldsymbol{w} \in \mathcal{Z}_n(L,d))$ since $Z_n(L,d) = 2^n \cdot Pr(\boldsymbol{w} \in \mathcal{Z}_n(L,d))$ and hence

$$\lim_{n\to\infty} \frac{\log(Z_n(L,d))}{n} = 1 + \lim_{n\to\infty} \frac{1}{n}\log(Pr(\boldsymbol{w} \in \mathcal{Z}_n(L,d))). \tag{6}$$

For integers $i,j \in [n-L+1]$ we notate $\boldsymbol{z} = (i,j)$ and denote by $I_{\boldsymbol{z}} = \mathbb{1}(\boldsymbol{w}_{i,L} \in \mathcal{B}_{d-1}(\boldsymbol{w}_{j,L}))$ the indicator function of the event that the Hamming distance of the L-substrings that start at positions i and j is less than d. We denote the set of all admissible pairs,

$$\Gamma = \{(i,j) \mid i,j \in [n-L+1], i<j\}.$$

Note that $|\Gamma| \leqslant n^2$. We are interested in a lower bound on

$$Pr(\boldsymbol{w} \in \mathcal{Z}_n(L,d)) = Pr\left(\sum_{\boldsymbol{z}\in\Gamma} I_{\boldsymbol{z}} = 0\right).$$

Note that for every $\boldsymbol{z} = (i,j) \in \Gamma$ it holds that

$$Pr(I_{\boldsymbol{z}}) = \frac{|\mathcal{B}_{d-1}(\boldsymbol{w}_{j,L})|}{2^L} \leqslant \frac{L^{d-1}+1}{2^L}.$$

Let $\boldsymbol{z} = (i,j), \boldsymbol{z}' = (i',j') \in \Gamma$. It is clear that if the substrings $\boldsymbol{w}_{i,L}, \boldsymbol{w}_{j,L}$ do not overlap with $\boldsymbol{w}_{i',L}$ or $\boldsymbol{w}_{j',L}$, then the indicators $I_{\boldsymbol{z}}, I_{\boldsymbol{z}'}$ are independent. We use Lemma 13 with a graph $G = (V,E)$ such that $V = \Gamma$ and there is an edge $\boldsymbol{z} \to \boldsymbol{z}'$ if at least one of $\boldsymbol{w}_{i,L}, \boldsymbol{w}_{j,L}$ overlaps with $\boldsymbol{w}_{i',L}$ or $\boldsymbol{w}_{j',L}$. It follows that every $\boldsymbol{z} \in \Gamma$ has at most $4Ln$ neighbors in the graph.

We set the numbers $\alpha_{\boldsymbol{z}} = \frac{1}{4Ln}$ for every $\boldsymbol{z} \in \Gamma$. Then, it holds that

$$\prod_{(\boldsymbol{z},\boldsymbol{z}')\in E} (1-\alpha_{\boldsymbol{z}'}) \geqslant \left(1 - \frac{1}{4Ln}\right)^{4Ln} \geqslant \frac{1}{e}$$

since the last expression approaches e^{-1} from above as $n \to \infty$. The condition of the lemma holds for every $\boldsymbol{z} \in \Gamma$ since

$$\begin{aligned} Pr(I_{\boldsymbol{z}}) &\leqslant \frac{L^{d-1}+1}{2^L} \leqslant \frac{L^{d-1}+1}{n^a} \overset{\text{(i)}}{\leqslant} \frac{1}{4Ln}\cdot\frac{1}{e} \\ &\leqslant \alpha_{\boldsymbol{z}} \prod_{(\boldsymbol{z},\boldsymbol{z}')\in E} (1-\alpha_{\boldsymbol{z}'}), \end{aligned}$$

where (i) follows from $4Ln = o(\frac{n^a}{L^{d-1}+1})$ since $L = \lceil a\log(n)\rceil$ and $a > 1$. By applying Lemma 13 we obtain

$$Pr(\boldsymbol{w} \in Z_n(L,d)) \geqslant \prod_{\boldsymbol{z}\in\Gamma} (1-\alpha_{\boldsymbol{z}}) \geqslant \left(1 - \frac{1}{4Ln}\right)^{n^2}.$$

Finally, since

$$\left(1 - \frac{1}{4Ln}\right)^{n^2} \approx \exp\left(-\frac{n}{4a\log(n)}\right),$$

it follows that $\frac{1}{n}\log(Pr(\boldsymbol{w} \in Z_n(L,d)))$ approaches 0 as $n \to \infty$. By plugging into (6) the theorem statement holds. ■

C. Encoding of (L,d)-Substring Distant Strings

In this section, a generic encoding algorithm is presented that uses a single redundancy bit in order to encode length-n strings that are (L,d)-substring distant, for

$$L = 2\log(n) + 2(d-1+\epsilon)\log(\log(n)),$$

where $\epsilon > 0$ and for n large enough. Note that this value of L is far from the value derived in Lemma 18 only by roughly $(d-1)\log(\log(n))$.

First, we present some helpful definitions. Let $\boldsymbol{w}, \boldsymbol{w}' \in \Sigma^n$ be strings such that $d_H(\boldsymbol{w},\boldsymbol{w}') \leqslant \rho$ for an integer $\rho \leqslant n$. The construction $EncDist_{n,\rho}(\boldsymbol{w},\boldsymbol{w}')$ is taken from [20] and encodes the distance between $\boldsymbol{w},\boldsymbol{w}'$. Let $p_1,\ldots,p_{d_H(\boldsymbol{w},\boldsymbol{w}')}$ denote the indices of the entries which $\boldsymbol{w},\boldsymbol{w}'$ do not agree upon. For every $i \in [\rho]$ let $\boldsymbol{y}_i \in \Sigma^{\log(n)}$ be the following value:

$$\boldsymbol{y}_i = \begin{cases} b(p_i) & i \leqslant d_H(\boldsymbol{w},\boldsymbol{w}') \\ 0^{\log(n)} & \text{Otherwise} \end{cases}$$

Thus,

$$EncDist_{n,\rho}(\boldsymbol{w},\boldsymbol{w}') = \boldsymbol{y}_1 \circ \cdots \circ \boldsymbol{y}_\rho.$$

Notice that the size of the output is independent of $\boldsymbol{w},\boldsymbol{w}'$ and equals $\rho\cdot\log(n)$. We sometimes omit the parameter n if it is clear from the context.

We utilize a marker substring, first introduced in [20], which we notate as a *d-auto cyclic string*. A string $\boldsymbol{u} \in \Sigma^n$ is a d-auto cyclic string, if it satisfies

$$d_H(\boldsymbol{u}, 0^i \circ \boldsymbol{u}_{1,n-i}) \geqslant d$$

for every $1 \leqslant i \leqslant d$. The authors of [20] also presented a construction of such strings of length $d\lceil\log(d)\rceil + 2d$. Let $\boldsymbol{u}_d$ denote a d-auto cyclic string for the rest of this section.

Next, let $\boldsymbol{w} \in \Sigma^k$ be a string for an integer $k \leqslant n$. Additionally, let $r \leqslant n$ be some integer. We want to construct a set of length-n strings, that contains all $\boldsymbol{y} \in \Sigma^n$ that satisfy

$$d_H(\text{Pref}_n(\boldsymbol{w}\circ\boldsymbol{y}),\boldsymbol{y}) \leqslant r. \tag{7}$$

Therefore, we construct the *concatenation ball of radius-r* around $\boldsymbol{w}$, denoted as $\mathcal{CB}_{n,r}(\boldsymbol{w})$. For this purpose, let $m = \lceil n/k\rceil$ and let $r_1,\ldots,r_m$ be a series of integers such that $\sum_{i=1}^m r_i \leqslant r$. Furthermore, let $\boldsymbol{w}_0,\boldsymbol{w}_1,\ldots,\boldsymbol{w}_m$ be a series of substrings such that $\boldsymbol{w}_0 = \boldsymbol{w}$ and for every $i \in [m]$, $\boldsymbol{w}_i \in \mathcal{B}_{r_i}(\boldsymbol{w}_{i-1})$. Thus, the string $\text{Pref}_n(\boldsymbol{w}_1\circ\cdots\circ\boldsymbol{w}_m)$ belongs to the set $\mathcal{CB}_{n,r}(\boldsymbol{w})$. Namely,

$$\mathcal{CB}_{n,r}(\boldsymbol{w}) = \left\{ \text{Pref}_n(\boldsymbol{w}_1\circ\cdots\circ\boldsymbol{w}_m) \;\middle|\; \begin{array}{l} \exists\{r_i\}_{i=1}^m \text{ s.t. } \sum_{i=1}^m r_i \leqslant r, \\ \boldsymbol{w}_0 = \boldsymbol{w} \text{ and } \forall i \in [m], \\ \boldsymbol{w}_i \in \mathcal{B}_{r_i}(\boldsymbol{w}_{i-1}) \end{array} \right\}.$$

One can verify that for every $\boldsymbol{y} \in \Sigma^n$ that satisfies (7), then $\boldsymbol{y} \in \mathcal{CB}_{n,r}(\boldsymbol{w})$.

Algorithm 4 receives a string $\boldsymbol{w} \in \Sigma^{n-1}$, and outputs a string $\boldsymbol{x} \in \mathcal{Z}_n(L,d)$. The algorithm shares ideas with the encoding scheme of *repeat-free words* from [12], and consists of two main procedures, elimination and expansion. First, we append to $\boldsymbol{w}$ a marker substring of length $L/2+d+1$ that contains the d-auto cyclic string $\boldsymbol{u}_d$, which is used by the decoder to identify the end of the information string. Then, at the elimination procedure we repeatedly look for substrings of length L that their Hamming distance is less than d. When found, we remove the first of the substrings and encode the occurrence using the function $EncDist_{L,d-1}$. Likewise, we eliminate occurrences of substrings of length $L/2$ that their Hamming distance from the $(L/2)$-suffix of the string is less than d. During this procedure, we ensure that the marker substring located at the suffix of the string remains intact. Later, at the expansion procedure we enlarge the string to length n by inserting substrings of length $L/2$ while making sure that the string remains (L,d)-substring distant. We denote for the rest of this section

$$\ell = L/2 = \log(n) + (d-1+\epsilon)\log(\log(n)).$$

We prove the correctness of the algorithm in the next few claims.

Claim 20: Algorithm 4 reaches Step 16, i.e. the elimination procedure terminates.

Proof: We prove by showing that at each iteration of the elimination loop, the length of $\boldsymbol{x}$ decreases. We analyze each case of removal and insertion independently. All length comparisons are taken for a large enough n.

Step 5: The minimal possible size of the removed substring at this step is achieved when $i = |\boldsymbol{x}| - L$. Thus, the algorithm removes a substring of length at least

$$L - \ell - d + 1 = \log(n) + (d-1+\epsilon)\log(\log(n)) - d + 1$$

and inserts a smaller substring of length

$$(d-1)\log(L) + 2\log(\ell+d) + 3.$$

Step 7: The algorithm removes a substring of length

$$L = 2\log(n) + 2(d-1+\epsilon)\log(\log(n)),$$

and inserts a substring of length

$$2\log(n) + (d-1)\log(L) + 3.$$

Step 11: The minimal possible length of the removed substring at this step is reached when $i = |\boldsymbol{x}| - 2\ell + |\boldsymbol{u}_d| - 1$. Therefore, the algorithm removes a substring of length at least

$$\ell - (|\boldsymbol{u}_d| + d + 1) = \\ \log(n) + (d-1+\epsilon)\log(\log(n)) - (|\boldsymbol{u}_d| + d + 1),$$

and inserts a smaller substring of length

$$(d-1)\log(\ell) + \log(|\boldsymbol{u}_d| + d + 1) + 2.$$

Step 13: The algorithm removes a substring of length

$$\ell = \log(n) + (d-1+\epsilon)\log(\log(n)),$$

and inserts a substring of length

$$\log(n) + (d-1)\log(\ell) + 1 = \\ \log(n) + (d-1)\log\big(\log(n) + (d-1+\epsilon)\log(\log(n))\big) + 1,$$

which is shorter for n large enough. ■

Claim 21: At Step 16 of Algorithm 4, the string $\boldsymbol{x}$

(1) is (L,d)-substring distant,
(2) ends with $0 \circ 1^d \circ 0^{\ell-|\boldsymbol{u}_d|} \circ \boldsymbol{u}_d$,
(3) contains no other ℓ-substring from $\mathcal{B}_{d-1}(\text{Suff}_\ell(\boldsymbol{x}))$ besides its ℓ-suffix.

Proof: Properties (1) and (2) follow immediately from the algorithm, since the loop continues as long as $\boldsymbol{x}$ is not (L,d)-distant, while ensuring that $\text{Suff}_{\ell+d+1}(\boldsymbol{x})$ is not touched.

As for (3), from (2) we have that $\text{Suff}_\ell(\boldsymbol{x}) = 0^{\ell-|\boldsymbol{u}_d|} \circ \boldsymbol{u}_d$ and from the definition of a d-auto cyclic string, for every $i \in [d]$,

$$d_H(\boldsymbol{u}_d, 0^i \circ (\boldsymbol{u}_d)_{1,|\boldsymbol{u}_d|-i-1}) \geqslant d$$

and hence for every $i \in [|\boldsymbol{x}| - \ell - d, |\boldsymbol{x}| - \ell]$ we have

$$d_H(\text{Suff}_\ell(\boldsymbol{x}), \boldsymbol{x}_{i,\ell}) \geqslant d.$$

For $i \in [|\boldsymbol{x}| - 2\ell + |\boldsymbol{u}_d|, |\boldsymbol{x}| - \ell - d - 1]$, the substring $\boldsymbol{x}_{i,\ell}$ has 1^d starting at position $(|\boldsymbol{x}| - \ell - d) - i$ while $\text{Suff}_\ell(\boldsymbol{x})$ has 0^d at this position. Other cases, for $i < |\boldsymbol{x}| - 2\ell + |\boldsymbol{u}_d|$, are eliminated at Step 9. ■

Claim 22: For every iteration of the expansion loop of Algorithm 4, the set $\left|\Sigma^\ell \setminus B\right|$ constructed in Step 18 is not empty.

Proof: Using simple counting arguments we have that for every $\boldsymbol{w} \in \Sigma^\ell$, the size of the radius-$(d-1)$ Hamming ball around $\boldsymbol{w}$ satisfies

$$|\mathcal{B}_{d-1}(\boldsymbol{w})| = \sum_{d'=0}^{d-1} \binom{\ell}{d'} \leqslant \ell^{d-1}.$$

Similarly, for every $\boldsymbol{w} \in \Sigma^k$ with $k \in [\ell]$, $\mathcal{CB}_{\ell,d-1}(\boldsymbol{w})$ can be bounded using the same value. Thus, the size of B is bounded by $n \cdot \ell^{d-1}$. It is left to show that $n \cdot \ell^{d-1} \leqslant |\Sigma^\ell| = 2^\ell$. By taking a logarithm from both sides of the equation, we derive that it is necessary that

$$\log(n) + (d-1)\log(\ell) \leqslant \ell$$

which is satisfied for n large enough by the value

$$\ell = \log(n) + (d-1+\epsilon)\log(\log(n)).$$

■

Let m denote the number of iterations of the expansion loop of Algorithm 4 that were executed. For every $k \in [m]$, let $\boldsymbol{x}_k$ denote the value of $\boldsymbol{x}$ at the end of the k-th iteration, and let $\boldsymbol{y}_k$ denote the string $\boldsymbol{y}$ that the algorithm picked at Step 19 of that iteration. We notate by $\boldsymbol{x}_0$ the value of $\boldsymbol{x}$ before the first iteration of the expansion loop. In the next two lemmas, when referring to $\boldsymbol{x}_k$ we sometimes omit the subscript k if it is clear from the context.

Claim 23: For every iteration $k \in [m]$, the string $\boldsymbol{x} = \boldsymbol{x}_{k-1} \circ \boldsymbol{y}_k$ satisfies that for every $i \in [1, |\boldsymbol{x}| - \ell]$,

$$d_H(\boldsymbol{x}_{i,\ell}, \boldsymbol{y}_k) \geqslant d.$$

```
Algorithm 4 LDEncode($\boldsymbol{w}, L, d$)
Input: A string $\boldsymbol{w} \in \Sigma^{n-1}$
Output: A string $\boldsymbol{x} \in \mathcal{Z}_n(L,d)$
 1: Set $\boldsymbol{x} = \boldsymbol{w} \circ 0 \circ 1^d \circ 0^{\ell-|\boldsymbol{u}_d|} \circ \boldsymbol{u}_d$
    Elimination:
 2: while exist indexes $i < j$ such that $d_H(\boldsymbol{x}_{i,L}, \boldsymbol{x}_{j,L}) < d$ or an index $i \leqslant |\boldsymbol{x}| - 2\ell + |\boldsymbol{u}_d|$ where $d_H(\boldsymbol{x}_{i,\ell}, 0^{\ell-|\boldsymbol{u}_d|} \circ \boldsymbol{u}_d) < d$
    do
 3:     case 1: violating substrings $\boldsymbol{x}_{i,L}, \boldsymbol{x}_{j,L}$ exists
 4:         if $i, j \in J_1 = [|\boldsymbol{x}| - L - \ell - d, |\boldsymbol{x}| - L + 1]$ ( $\boldsymbol{x}_{i,L}$ intersects with the suffix $0 \circ 1^d \circ 0^{\ell-|\boldsymbol{u}_d|} \circ \boldsymbol{u}_d$) then
 5:             Remove $\boldsymbol{x}_{i,L-(|\boldsymbol{x}|-L-\ell-d-i)+1}$, append $100 \circ b_{J_1}(i) \circ b_{J_1}(j) \circ EncDist_{L,d-1}(\boldsymbol{x}_{i,L}, \boldsymbol{x}_{j,L})$ to the left of $\boldsymbol{x}$
 6:         else
 7:             Remove $\boldsymbol{x}_{i,L}$, append $101 \circ b(i) \circ b(j) \circ EncDist_{L,d-1}(\boldsymbol{x}_{i,L}, \boldsymbol{x}_{j,L})$ to the left of $\boldsymbol{x}$
 8:         end if
 9:     case 2: a substring $\boldsymbol{x}_{i,\ell}$ with $i < |\boldsymbol{x}| - 2\ell + |\boldsymbol{u}_d|$ such that $d_H(\boldsymbol{x}_{i,\ell}, 0^{\ell-|\boldsymbol{u}_d|} \circ \boldsymbol{u}_d) < d$ exists
10:         if $i \in J_2 = [|\boldsymbol{x}| - 2\ell - d, |\boldsymbol{x}| - 2\ell + |\boldsymbol{u}_d| - 1]$ ( $\boldsymbol{x}_{i,\ell}$ intersects with the suffix $0 \circ 1^d \circ 0^{\ell-|\boldsymbol{u}_d|} \circ \boldsymbol{u}_d$) then
11:             Remove $\boldsymbol{x}_{i,\ell-(|\boldsymbol{x}|-2\ell-d-i)+1}$, append $11 \circ b_{J_2}(i) \circ EncDist_{\ell,d-1}(\boldsymbol{x}_{i,\ell}, 0^{\ell-|\boldsymbol{u}_d|} \circ \boldsymbol{u}_d)$ to the left of $\boldsymbol{x}$
12:         else
13:             Remove $\boldsymbol{x}_{i,\ell}$, append $0 \circ b(i) \circ EncDist_{\ell,d-1}(\boldsymbol{x}_{i,\ell}, 0^{\ell-|\boldsymbol{u}_d|} \circ \boldsymbol{u}_d)$ to the left of $\boldsymbol{x}$
14:         end if
15: end while
16: if $|\boldsymbol{x}| \geqslant n$, return $\boldsymbol{x}_{1,n}$
    Expansion:
17: while $|\boldsymbol{x}| < n$ do
18:     Set
        $$B = \left( \bigcup_{i \in [1, |\boldsymbol{x}| - \ell]} \mathcal{B}_{d-1}(\boldsymbol{x}_{i,\ell}) \right) \cup \left( \bigcup_{i \in [|\boldsymbol{x}| - \ell + 1, |\boldsymbol{x}|]} \mathcal{CB}_{\ell,d-1}(\boldsymbol{x}_{i,|\boldsymbol{x}|-i+1}) \right)$$
19:     Pick $\boldsymbol{y} \in \Sigma^\ell \setminus B$ and append $\boldsymbol{x} = \boldsymbol{x} \circ \boldsymbol{y}$
20: end while
21: Return $\boldsymbol{x}_{1,n}$
```

Proof: According to the construction of B, for every $i \in [1, |\boldsymbol{x}_{k-1}| - \ell + 1]$ the ball $\mathcal{B}_{d-1}(\boldsymbol{x}_{i,\ell})$ is contained in B and since $\boldsymbol{y}_k \notin B$ then $d_H(\boldsymbol{x}_{i,\ell}, \boldsymbol{y}_k) \geqslant d$. Otherwise, let $i \in [|\boldsymbol{x}_{k-1}| - \ell + 2, |\boldsymbol{x}_{k-1}|]$, assume in the contrary that $d_H(\boldsymbol{x}_{i,\ell}, \boldsymbol{y}_k) < d$ and thus

$$d_H(\text{Pref}_\ell(\boldsymbol{x}_{i,|\boldsymbol{x}|-i+1} \circ \boldsymbol{y}_k), \boldsymbol{y}_k) < d.$$

However, it follows that $\boldsymbol{y}_k \in \mathcal{CB}_{\ell,d-1}(\boldsymbol{x}_{i,|\boldsymbol{x}|-i+1})$ which is a contradiction. Since $|\boldsymbol{x}| - \ell = |\boldsymbol{x}_{k-1}|$, this concludes the proof. ■

Claim 24: For every iteration $k \in [m]$, the string $\boldsymbol{x}_k$ is (L,d)-substring distant.

Proof: We prove the lemma by induction over the values of k. For the base case $k = 1$, let $\boldsymbol{x} = \boldsymbol{x}_0 \circ \boldsymbol{y}_1$ and assume in the contrary that there are two substrings $\boldsymbol{x}_{i,L}, \boldsymbol{x}_{j,L}$ of Hamming distance less than d. Since $\boldsymbol{x}_0$ is (L,d)-substring distant from Claim 21 Statement (1), we only need to consider the cases where $\boldsymbol{x}_{j,L}$ overlaps with $\boldsymbol{y}_1$. Therefore, using lengths considerations, $\boldsymbol{x}_{j,L}$ contains $\text{Suff}_\ell(\boldsymbol{x}_0) = 0^{\ell-|\boldsymbol{u}_d|} \circ \boldsymbol{u}_d$ at some position $r \in [\ell]$. It follows that $d_H(\boldsymbol{x}_{i+r,\ell}, \text{Suff}_\ell(\boldsymbol{x}_0)) < d$ which contradicts Claim 21 Statement (3).

Next, we assume the lemma holds for $\boldsymbol{x}_{k-1}$ with $k \geqslant 1$ and prove its correctness for $\boldsymbol{x} = \boldsymbol{x}_{k-1} \circ \boldsymbol{y}_k$. Assume in the contrary that $\boldsymbol{x}_{i,L}, \boldsymbol{x}_{j,L}$ satisfy $d_H(\boldsymbol{x}_{i,L}, \boldsymbol{x}_{j,L}) < d$. Using the induction assumption, we only need to consider the values of i, j where $\boldsymbol{x}_{j,L}$ overlaps with $\boldsymbol{y}_k$. Thus, it follows that $\boldsymbol{x}_{j,L}$ contains the substring $\boldsymbol{y}_{k-1}$, at some position $r \in [\ell]$. However, this implies that $\boldsymbol{x}_{i+r,\ell}$ is a substring of $\boldsymbol{x}_k$ that satisfies $d(\boldsymbol{x}_{i+r,\ell}, \boldsymbol{y}_k) < d$ while $i + r \leqslant |\boldsymbol{x}| - \ell$ which is a contradiction to Claim 23. ■

Theorem 25: Algorithm 4 successfully returns a string from $\mathcal{Z}_n(L,d)$.

Proof: If the condition in Step 16 holds then according to Claim 21 Statement (1), $\boldsymbol{x}$ is (L,d)-substring distant. Since every substring of $\boldsymbol{x}$ is (L,d)-substring distant as well, the algorithm returns in this case a string that belongs to $\mathcal{Z}_n(L,d)$. Otherwise, from Claim 24, the algorithm returns a (L,d)-substring distant string of length n at Step 21. ■

The decoding scheme receives $\boldsymbol{x}$ which is an output of Algorithm 4 and outputs $\boldsymbol{w} \in \Sigma^{n-1}$. First, we look for the leftmost occurrence of the substring $\boldsymbol{v} = 0 \circ 1^d \circ 0^{\ell-|\boldsymbol{u}_d|} \circ \boldsymbol{u}_d$ in $\boldsymbol{x}$. According to Claim 21 Statement (2), the part of the string to the right of this substring was added during the expansion procedure and therefore we remove it from $\boldsymbol{x}$. If the substring $\boldsymbol{v}$ is not present, we look for its longest prefix that is located as a suffix of $\boldsymbol{x}$. The substring we found is a part of the substring $\boldsymbol{v}$ the algorithm added at Step 1 since the output of the algorithm is longer than the input. Thus, we can complete the substring to $\boldsymbol{v}$ and receive $\boldsymbol{x}$ at the stage of after the elimination procedure.

Next, we iteratively inverse the elimination procedure. We identify using the first three entries of $\boldsymbol{x}$ the last step at which the data was encoded. If we encoded the data at Step 5 or Step 7, we decode i, j from the function b, and recover $\boldsymbol{x}_{j,L}$ using $\boldsymbol{x}_{i,L}$ and the encoded distance. If $j \leqslant i+\ell$ this has to be done carefully, by restoring every $j-i$ entries of $\boldsymbol{x}_{j,L}$ a time. If we encoded the data at Step 11 or Step 13, we decode from the outputs of the functions b and $EncDist_{L,d-1}$ the position i and the substring $\boldsymbol{x}_{i,\ell}$, and insert the substring at position i. We repeat this process until we obtain a substring of length $n+\ell+d$, and return its $(n-1)$-prefix as $\boldsymbol{w}$.

V. Alternative Construction for Erroneous Multispectrum

One of the benefits of the construction in Section IV is that it can tolerate a relatively large number of erroneous substrings. Namely, t can have any value less than $L/2$. However, at the same time, the construction is identical for small and large values of t. In this section, we show an alternative construction for moderate values of t. More specifically, for $t \leqslant \log(n)/\log(\log(n))$ we will have a weaker constraint than the one stated in Theorem 16. This will be given by strings that satisfy several constraints that are presented in Definition 26, along with a reconstruction algorithm, presented in Algorithm 5. One difference between the algorithm presented in this section and the one from Section IV is that the outcome here is not necessarily a length-n string. However, the length of the output decoded string is at least $n-4t$ and it will be shown that it contains the substring $\boldsymbol{x}_{2t+1,n-4t}$ with no error.

For the rest of this section, similarly to Section III, we again use the integers $\ell_1 = L - \lfloor t/3 \rfloor - 1, \ell_2 = L - \lceil 2t/3 \rceil - 1, \ell_3 = L - t - 1$ and the sets $I_2 = [n-\ell_2-t+1, n-\ell_2+1], I_3 = [n-\ell_3-t+1, n-\ell_3+1]$.

Definition 26: A string $\boldsymbol{x} \in \Sigma^n$ is said to satisfy the (n, L, t, s)**-erroneous reconstruction (EREC) constraints** if it fulfills the following three constraints.

1) $\boldsymbol{x}$ is a $(\ell_1, 2s+1)$-substring distant string.
2) The first and last $t+1$ length-ℓ_2 substrings have a Hamming distance of at least $2s+1$ from all other length-ℓ_2 substrings. Namely, for all $i \in [t+1], j \in [n-\ell_2+1]$ with $i \neq j$ then $d_H(\boldsymbol{x}_{i,\ell_2}, \boldsymbol{x}_{j,\ell_2}) \geqslant 2s+1$ and for all $i \in [n-\ell_2+1], j \in I_2$ with $i \neq j$, then $d_H(\boldsymbol{x}_{i,\ell_2}, \boldsymbol{x}_{j,\ell_2}) \geqslant 2s+1$.
3) The first $t+1$ length-ℓ_3 substrings have a Hamming distance of at least $2s+1$ from the last $t+1$ length-ℓ_3 substrings. Namely, for all $i \in [t+1], j \in I_3$, $d_H(\boldsymbol{x}_{i,\ell_3}, \boldsymbol{x}_{j,\ell_3}) \geqslant 2s+1$.

Notice that these constraints are a variation of the constraints presented in Definition 5, with the demand for inequality replaced with a demand of Hamming distance of at least $2s+1$. In fact, when applying $s=1$ we receive that the $(n, L, t, 1)$-EREC constraints are exactly the (n, L, t)-LREC constraints. For n, L, t, s, denote by $\mathcal{D}_n(L, t, s)$ the set of all strings that satisfy the (n, L, t, s)-EREC constraint and let $D_n(L, t, s) = |\mathcal{D}_n(L, t, s)|$.

Let $U \in \mathcal{B}_{L,t,s}(\boldsymbol{x})$ for $\boldsymbol{x} \in \Sigma^n$. Next, we modify the definition of the maximal-reconstructible string of U, and denote it as $\mathbf{W}_3(U)$. Let $V = \{\boldsymbol{u}_{i_1}, \ldots, \boldsymbol{u}_{i_m}\} \subseteq U$ denote for the rest of this section a subset of U where $m \geqslant n-t$. Let $I(V) = [i_1, i_m + L - 1]$ be the set of positions of $\boldsymbol{x}$ entries that appear in V (the segment is continuous since $t < L$), and let $n(V) = i_m + L - i_1$ denote its size. We define for every $j \in I(V)$, the function $\text{cons}(V, j)$ which returns the consented value of the entry x_j in V, or an error if such a value does not exist. That is,

$$\text{cons}(V,j) = \begin{cases} a & \text{if for every } k \in [m] \text{, where} \\ & j \in [i_k, i_k+L-1], (\boldsymbol{u}_{i_k})_{j-i_k+1} = a \\ error & \text{otherwise} \end{cases}.$$

We say that V has a *consensus* if for every $j \in I(V)$, $\text{cons}(V, j) \neq error$. For every such V we define its consented string to be

$$\text{cons}(V) = (w_1, \ldots, w_{n(V)}) \text{ with } w_i = cons(V, i+i_1-1).$$

Finally, Let $V^* \subseteq U$ denote the largest subset of U that has a consensus. Thus, we define $\mathbf{W}_3(U) = \text{cons}(V^*)$. For simplicity, we say that V^* is the subset of U that contains only its correct substrings. Namely,

$$V^* = \{\boldsymbol{u}_i \in U \mid i \notin I_e(U)\}.$$

In exceptional cases V^* might be a different set and we will analyze those cases later in this section.

We modify the definition of (L, t, s)-reconstructible strings in Definition 15 to use $\mathbf{W}_3(U)$ instead of $\mathbf{W}_2(U)$. The main theorem of this subsection is presented next, in Theorem 27. The proof of this theorem is given by an explicit reconstruction algorithm.

Theorem 27: Every string $\boldsymbol{x} \in \mathcal{D}_n(L, t, s)$ is an (L, t, s)-reconstructible string.

Algorithm 5 uses the stitching and reconstruction algorithms presented in Section III. First, we extend $\text{Reconstruct}(U, t)$ to notice errors by appending an additional step at its end. If $\text{Reconstruct}(U, t)$ reaches this step, it returns $error$. It follows that such a result occurs if $\text{Reconstruct}(U, t)$ fails to successfully reconstruct at Steps 2, 3 and 4. Algorithm 5 receives $U \in \mathcal{B}_{L,t,s}(\boldsymbol{x})$ for $\boldsymbol{x} \in \mathcal{D}_n(L, t, s)$ and reconstructs $\mathbf{W}_3(U)$. First, it invokes the stitching algorithm with U and $t' = 0$ and receives a set of substrings denoted by $A_0 = \{\boldsymbol{y}_1, \ldots, \boldsymbol{y}_r\}$. The fact that $\boldsymbol{x}$ is $(L - \lfloor t/3 \rfloor - 1, 2s+1)$-substring distant (and in particular $(L-1, 2s+1)$-substring distant) ensures that errors are present only at the edges of the substrings of $\boldsymbol{y}_1, \ldots, \boldsymbol{y}_r$. Based on this observation, we look for a subset of U with maximal size that has a consensus. For every candidate spectrum V, we use $\text{Reconstruct}(V, t)$ to identify if V has a consensus, and return the consented string.

Before presenting the algorithm, we define for a string $\boldsymbol{w} \in \Sigma^n$ and an integer $k \in [n]$, its *striping ball* of radius k, denoted by $SB_k(\boldsymbol{w})$, as the set

$$SB_k(\boldsymbol{w}) = \{\boldsymbol{w}_{i+1, n-j-i+1} \mid 0 \leqslant i+j \leqslant k\}.$$

Algorithm 5 (n, L, t, s)-EREC Reconstruction

```
Input: $U \in \mathcal{B}_{L,t,s}(\boldsymbol{x})$ for $\boldsymbol{x} \in \mathcal{D}_n(L, t, s)$
Output: $\mathbf{W}_3(U)$ the maximum reconstructible-substring of $U$
1: $A_0 = \text{Stitch}(U, 0)$, denote $A_0 = \{\boldsymbol{y}_1, \ldots, \boldsymbol{y}_r\}$
2: Set $\alpha = t - \lceil (r-1)/2 \rceil + 1$, and construct
     $B = \{\bigcup_{j=1}^{r} S_L(\boldsymbol{w}_j) \mid \forall_{j \in [r]} : \boldsymbol{w}_j \in SB_\alpha(\boldsymbol{y}_j)\},$
     $\forall \rho \in [0, t]$ set $B_\rho = \{V \in B \mid |V| = |U| - \rho\}.$
3: for every $\rho \in [0, t]$ do
4:     if exists $V \in B_\rho$ with Reconstruct$(V, t) \neq error$ then
5:         return Reconstruct$(V, t)$.
6:     end if
7: end for
```

We prove the correctness of Algorithm 5 in the next few claims. Let $\boldsymbol{u}_1, \ldots, \boldsymbol{u}_{n-L+1}$ denote the elements of U with respect to the order of $S_L(\boldsymbol{x})$.

Claim 28: At Step 1, in Stitch$(U, 0)$ we stitch two substrings $\boldsymbol{w}, \boldsymbol{w}'$ if $\text{Suff}_L(\boldsymbol{w}) = \boldsymbol{u}_i$ and $\text{Pref}_L(\boldsymbol{w}') = \boldsymbol{u}_{i+1}$ for some $\boldsymbol{u}_i, \boldsymbol{u}_{i+1} \in U$.

Proof: Let $\boldsymbol{w}, \boldsymbol{w}'$ denote two substrings with $\text{Suff}_L(\boldsymbol{w}) = \boldsymbol{u}_i$ and $\text{Pref}_L(\boldsymbol{w}') = \boldsymbol{u}_j$ that were stitched in Stitch$(U, 0)$ using $\text{Suff}_{L-1}(\boldsymbol{w}) = \text{Pref}_{L-1}(\boldsymbol{w}')$. Assume in the contrary that $j \neq i+1$. From (n, L, t, s)-EREC Constraint 1 it follows that $\boldsymbol{x}$ is $(L-1, 2s+1)$-substring distant. Thus, since $U \in \mathcal{B}_{L,t,s}(\boldsymbol{x})$ we have that

$$
\begin{aligned}
&d_H(\text{Suff}_{L-1}(\boldsymbol{u}_i), \text{Pref}_{L-1}(\boldsymbol{u}_j)) \\
&\quad \geqslant d_H(\boldsymbol{x}_{i+1,L-1}, \boldsymbol{x}_{j,L-1}) - d_H(\text{Suff}_{L-1}(\boldsymbol{u}_i), \boldsymbol{x}_{i+1,L-1}) \\
&\quad\quad - d_H(\text{Pref}_{L-1}(\boldsymbol{u}_j), \boldsymbol{x}_{j,L-1}) \\
&\quad \geqslant 2s+1-s-s = 1.
\end{aligned}
$$

which is a contradiction. ■

For every substring $\boldsymbol{y}_j \in A_0$, we notate by Y_j the *substring-set* of $\boldsymbol{y}_j$ which is the set that contains all the substrings of U that are substrings of $\boldsymbol{y}_j$. That is, $Y_j = S_L(\boldsymbol{y}_j) = \{\boldsymbol{u}_{i_j}, \ldots, \boldsymbol{u}_{i_j+m_j}\}$ where $m_j = |\boldsymbol{y}_j| - L$. We say that Y_j is erroneous if it contains an erroneous substring.

Claim 29: After Step 1, the following conditions hold.

(1) There are at least $\lceil (r-1)/2 \rceil$ substrings in A_0 that their substring-set contains an erroneous substring,
(2) $r \leqslant 2t+1$,
(3) For every $j \in [r]$, the substring-set Y_j contains at most $\alpha = t - \lceil (r-1)/2 \rceil + 1$ erroneous substrings.

Proof: Assume that $\boldsymbol{y}_1, \ldots, \boldsymbol{y}_r$ are ordered in correspondence to the positions of their substring-sets; that is, $i_1 = 1, i_r + m_r = n - L + 1$ and for every $j \in [2, r]$, $i_{j-1} + m_{j-1} + 1 = i_j$. Thus, using Claim 28, for every $j \in [1, r-1]$, at least one of $\boldsymbol{u}_{i_j+m_j}, \boldsymbol{u}_{i_{j+1}}$ is erroneous. Therefore, the minimal number of erroneous substring-sets is $\lceil (r-1)/2 \rceil$. It can be achieved for example when only the substring-sets with even indecies are erroneous (and additionally Y_r if r is odd), thus proving statement (1). Statements (2) and (3) follows immediately from (1) and from the fact that there are at most t erroneous substrings, and thus at most t erroneous substring-sets. ■

Claim 30: For every $\boldsymbol{y}_j \in A_0$ there exists a substring $\boldsymbol{w}_j \in SB_\alpha(\boldsymbol{y}_j)$ such that $S_L(\boldsymbol{w}_j)$ contains only the correct substrings of Y_j.

Proof: First, if Y_j contains no erroneous substrings, picking $\boldsymbol{w}_j = \boldsymbol{y}_j$ yields the required result. Else, let g, h denote the positions of the erroneous entries of $\boldsymbol{y}_j$ that are closest to the center of the substring, from the left and right receptively. Namely, denote by $I_e(\boldsymbol{y}_j)$ the positions of the erroneous entries of $\boldsymbol{y}_j$, so we get that

$$
g = \max\left\{ i \in I_e(\boldsymbol{y}_j) \mid i \leqslant \left\lfloor \frac{|\boldsymbol{y}_j|}{2} \right\rfloor \right\},
$$
$$
h = \min\left\{ i \in I_e(\boldsymbol{y}_j) \mid i \geqslant \left\lceil \frac{|\boldsymbol{y}_j|}{2} \right\rceil \right\}.
$$

Assume without loss of generality that both g, h exist. From Claim 28, all the substrings of Y_j that contain those entries agree with the errors. Therefore, the sets $E(g) = Y_j \cap \{\boldsymbol{u}_{g-L+1}, \ldots, \boldsymbol{u}_g\}, E(H) = Y_j \cap \{\boldsymbol{u}_{h-L+1}, \ldots, \boldsymbol{u}_h\}$ contain the erroneous entries $(\boldsymbol{y}_j)_g, (\boldsymbol{y}_j)_h$, respectively. Since from Claim 29 statement (3), Y_j contains at most α erroneous substrings and since $k < L$, we have

$$
E(g) = \{\boldsymbol{u}_{i_j}, \ldots, \boldsymbol{u}_g\}, E(h) = \{\boldsymbol{u}_{h-L+1}, \ldots, \boldsymbol{u}_{i_j+m_j}\}
$$

and hence $|E(g)| + |E(h)| \leqslant \alpha$. Finally, we can pick $\boldsymbol{w}_j = (w_{g+1}, \ldots, w_{h-L})$ which satisfies $|\boldsymbol{w}_j| \leqslant |\boldsymbol{y}_j| - k$ or alternatively $\boldsymbol{w}_j \in SB_\alpha(\boldsymbol{y}_j)$. It is ensured from the selection of g, h that $\boldsymbol{w}_j$ is error-free. ■

Let B^* denote the union $\bigcup_{\rho=0}^{t} B_\rho$. Notice that every spectrum $V \in B^*$ satisfies $V \subseteq U$ and $|V| \geqslant |U| - t$.

Claim 31: For every $V \in B^*$, Reconstruct(V, t) returns $error$ if and only if there exist $i \in I(V)$ such that $\text{cons}(V, i) = error$.

Proof: In the proof of the reconstruction algorithm in Lemma 11 in Section III, we use arguments that rely on an (n, L, t)-LREC Constraint to justify that we stitched at some iteration k of the stitching algorithm two substrings $\boldsymbol{w}, \boldsymbol{w}'$ if $\text{Suff}_L(\boldsymbol{w}) = \boldsymbol{u}_i$ and $\text{Pref}_L(\boldsymbol{w}') = \boldsymbol{u}_{i+k}$. We can replace those arguments with arguments that are based on (n, L, t, s)-Constraint like we used in the proof of Claim 28, to prove the same claim for the operation of Reconstruct(V, t).

Thus, we can derive that Reconstruct(V, t) returned a successful result if and only if all the substrings of V were stitched in the correct order. Following previous observation, it is only possible if for every $i \in I(V)$, $\text{cons}(V, i) \neq error$ and in this case the result string is cons(V). ■

Claim 32: The set $V^* = \{\boldsymbol{u}_i \in U \mid i \notin I_e(U)\}$ belongs to B^* and Reconstruct(V^*, t) is successful.

Proof: Following Claim 30, there exists for every $\boldsymbol{y}_j \in A_0$ a substring $\boldsymbol{w}_j \in SB_\alpha(\boldsymbol{y}_j)$ such that $S_L(\boldsymbol{w}_j)$ contains only the correct substrings of Y_j. By picking such $\boldsymbol{w}_j$ for all substrings of A_0 that contained errors, we construct the set V^*. From the definition of B in Step 2, $V^* \in B$. Moreover, since there are at most t erroneous substrings, $V^* \in B^*$ as well. Since V^* contains no erroneous entries, it satisfies the conditions of Claim 31 and thus Reconstruct(V^*, t) is successful. ■

Claim 33: The size of the set B^* satisfies $|B^*| \leqslant n$

Proof: For every $\boldsymbol{y}_j \in A_0$, the size of $SB_\alpha(\boldsymbol{y}_j)$ can be a viewed as the number of possible selections of $n_{j,1}, n_{j,2}$ entries to remove from the left, right of $\boldsymbol{y}_j$ respectively, with $n_{j,1} + n_{j,2} \leqslant \alpha$. This reflects removing $n_{j,1} + n_{j,2}$ substrings from Y_j, which are also absent from a set $V \in B$. Therefore, for every $p \leqslant t$ we can bound the size of B_p with the number of possible solutions to the equation

$$\sum_{j=1}^{r} (n_{j,1} + n_{j,2}) = p,$$

where for every $j \in [r]$, $n_{j,1} + n_{j,2} \leqslant \alpha$. This value can be bounded using

$$|B_p| \leqslant \binom{p + 2r - 1}{p},$$

and therefore

$$|B^*| \leqslant \sum_{p=0}^{t} |B_p| \leqslant \sum_{p=0}^{t} \binom{5t+1}{p} \leqslant (5t+2)^t.$$

Since $t \leqslant \log(n)/\log(\log(n))$, for a large enough n we have that $|B^*| \leqslant n$. ■

Lastly, we conclude with the following lemma.

Lemma 34: Algorithm 5 returns $\mathbf{W}_3(U)$ and has a polynomial-time complexity.

Proof: If follows from Claim 32 that the algorithm finds $V^* \in B_\rho$ for $\rho = |I_e(U)|$. Thus, the algorithm returns cons$(V^*) = \mathbf{W}_3(U)$ at step 5.

The time complexity of the algorithm is dominated by the loop of Step 3. Since for any set $V \subseteq U$ the time complexity of Reconstruct (V, t) is a polynomial of n, and from Claim 33 the size of B^* is a polynomial of n, Algorithm 5 has polynomial-time complexity. ■

The proof of Lemma 34 also completes the proof of Theorem 27.

Finally, we analyze the results of Algorithm 5 in cases where $V^* \neq \{\boldsymbol{u}_i \in U \mid i \notin I_e(U)\}$. An erroneous substring can belong to V^* when the majority value of an entry is not the correct entry. This can occur if an entry appears erroneously in the majority of their occurrences in U, or in a subset of U after removal of some incorrect substrings. By addressing to the majority values of entries rather than their correct values in the proof of Algorithm 5, we can derive that the algorithm constructs $V^* \in B^*$ in Step 2 and thus reconstructs $\mathbf{W}_3(U)$ in this case as well. Since $t < L/2$ and $|V^*| \geqslant |U| - t$, $|\mathbf{W}_3(U)| \geqslant n - t$. Furthermore, since there are at most t erroneous substrings in V^*, for every $i \in [2t+1, n-2t]$, cons$(V^*, i) = \boldsymbol{x}_i$. Thus,

$$\mathbf{W}_3(U)_{i_1+2t+1, n-4t} = \boldsymbol{x}_{2t+1, n-4t},$$

where i_1 is the position of the first entry of $\boldsymbol{x}$ that appears in V^*.

VI. Conclusion

This paper studied the reconstruction of strings based upon noisy versions of their multispectrum. In the first model, we assumed that not all substrings in the multispectrum are read and in the second, it was assumed that all substrings are read, however several of them can be erroneous. In each case we studied code constructions of strings that can be uniquely reconstructed from the noisy version of the multispectrum. The cardinalities of the codes are studied along with specific code constructions. An important ingredient in our constructions is the set of (L, d)-substring distant strings. We studied when the redundancy of this set is at most a single bit and when its asymptotic rate approaches 1. We also presented specific encoding and decoding maps for this constraint. While this work studied only the binary case, most of the results in the paper can be extended for the non-binary case as well. While the results in the paper provide a significant contribution in the area of coding for reconstruction from substrings spectrum, there are still several interesting problems which are left open. Some of them are the constructions and analysis of more cases for the values of L, t, and d, and especially studying the erroneous case of edit errors together with losses of substrings.

Acknowledgment

The authors would like to thank the Associate Editor Amin Gohari and the three anonymous reviewers for their very helpful comments. They are also thankful to Ohad Elishco for helpful discussions on the proofs of Theorem 12 and Theorem 19.

References

[1] J. Acharya, H. Das, O. Milenkovic, A. Orlitsky, and S. Pan, “On reconstructing a string from its substring compositions,” in *Proc. IEEE Int. Symp. Inf. Theory*, Austin, TX, USA, Jun. 2010, pp. 1238–1242.

[2] J. Acharya, H. Das, O. Milenkovic, A. Orlitsky, and S. Pan, “String reconstruction from substring compositions,” *SIAM J. Discrete Math.*, vol. 29, no. 3, pp. 1340–1371, Jan. 2015.

[3] N. Alon and J. Spencer, *The Probabilistic Method*, 2nd ed. Wiley, 2000.

[4] R. Arratia, D. Martin, G. Reinert, and M. S. Waterman, “Poisson process approximation for sequence repeats, and sequencing by hybridization,” *J. Comput. Biol.*, vol. 3, no. 3, pp. 425–463, Jan. 1996.

[5] T. Batu, S. Kannan, S. Khanna, and A. McGregor, “Reconstructing strings from random traces,” in *Proc. 15th Annu. ACM-SIAM Symp. Discrete Algorithms*, New Orleans, LA, USA, 2004, pp. 910–918.

[6] 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 Support Program. Lang. Operating Syst.*, Atlanta, Georgia, Mar. 2016, pp. 637–649.

[7] G. Bresler, M. Bresler, and D. Tse, “Optimal assembly for high throughput shotgun sequencing,” *BMC Bioinf.*, vol. 14, no. S5, pp. 1–13, Apr. 2013.

[8] Z. Chang, J. Chrisnata, M. F. Ezerman, and H. M. Kiah, “Rates of DNA sequence profiles for practical values of read lengths,” *IEEE Trans. Inf. Theory*, vol. 63, no. 11, pp. 7166–7177, Nov. 2017.

[9] C.-S. Chin *et al.*, “Nonhybrid, finished microbial genome assemblies from long-read SMRT sequencing data,” *Nature Methods*, vol. 10, no. 6, pp. 563–569, Jun. 2013.

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

[11] M. Dudık and L. J. Schulman, “Reconstruction from subsequences,” *J. Combinat. Theory A*, vol. 103, no. 2, pp. 337–348, Aug. 2003.

[12] O. Elishco, R. Gabrys, M. Mèdard, and E. Yaakobi, “Repeat-free codes,” in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Paris, France, Jul. 2019, pp. 932–936.

[13] O. Elishco, R. Gabrys, M. Mèdard, and E. Yaakobi, "Repeat free codes," submitted for publication.

[14] P. Erdös and L. Lovász, "Problems and results on 3-chromatic hypergraphs and some related questions," in *Infinite and Finite Sets*, vol. 2, A. Hajnal, R. Rado, and V. T. Sós, Eds. Amsterdam, The Netherlands: North-Holland, 1975, pp. 609–627.

[15] R. Gabrys and O. Milenkovic, "Unique reconstruction of coded sequences from multiset substring spectra," *IEEE Trans. Inf. Theory*, vol. 65, no. 12, pp. 7682–7696, Jun. 2019.

[16] S. Ganguly, E. Mossel, and M. Z. Racz, "Sequence assembly from corrupted shotgun reads," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Barcelona, Spain, Jul. 2016, pp. 265–269.

[17] N. Goldman *et al.*, "Towards practical, high-capacity, low-maintenance information storage in synthesized DNA," *Nature*, vol. 494, no. 7435, pp. 77–80, Jan. 2013.

[18] 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.

[19] V. I. Levenshtein, "Efficient reconstruction of sequences from their subsequences or supersequences," *J. Combinat. Theory A*, vol. 93, no. 2, pp. 310–332, Feb. 2001.

[20] M. Levy and E. Yaakobi, "Mutually uncorrelated codes for DNA storage," *IEEE Trans. Inf. Theory*, vol. 65, no. 6, pp. 3671–3691, Jun. 2019.

[21] H. Li, "Minimap and miniasm: Fast mapping and de novo assembly for noisy long sequences," *Bioinformatics*, vol. 32, no. 14, pp. 2103–2110, Jul. 2016.

[22] N. Loman, J. Quick, and J. Simpson, "A complete bacterial genome assembled de novo using only nanopore sequencing data," *Nature Methods*, vol. 12, no. 8, pp. 733–735, 2015.

[23] B. Manvel, A. Meyerowitz, A. Schwenk, K. Smith, and P. Stockmeyer, "Reconstruction of sequences," *Discrete Math.*, vol. 94, no. 3, pp. 209–219, 1991.

[24] S. Marcovich and E. Yaakobi, "Reconstruction of strings from their substrings spectrum," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Los Angeles, CA, USA, Jun. 2020, pp. 658–663.

[25] A. S. Motahari, G. Bresler, and D. N. C. Tse, "Information theory of DNA shotgun sequencing," *IEEE Trans. Inf. Theory*, vol. 59, no. 10, pp. 6273–6289, Oct. 2013.

[26] A. Motahari, K. Ramchandran, D. Tse, and N. Ma, "Optimal DNA shotgun sequencing: Noisy reads are as good as noiseless reads," in *Proc. IEEE Int. Symp. Inf. Theory*, Istanbul, Turkey, Jul. 2013, pp. 1640–1644.

[27] O. Sabary, Y. Orlev, R. Shafira, L. Anavy, E. Yaakobi, and Z. Yakhini, "SOLQC: Synthetic oligo library quality control tool," *Bioinformatics*, vol. 37, no. 5, pp.720–722, 2020.

[28] S. L. Salzberg, "Mind the gaps," *Nature Methods*, vol. 7, no. 2, pp. 105–106, 2010.

[29] A. D. Scott, "Reconstructing sequences," *Discrete Math.*, vol. 175, nos. 1–3, pp. 231–238, Oct. 1997.

[30] I. Shomorony, T. Courtade, and D. Tse, "Do read errors matter for genome assembly?" in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Hong Kong, Jun. 2015, pp. 919–923.

[31] I. Shomorony, G. M. Kamath, F. Xia, T. A. Courtade, and D. N. Tse, "Partial DNA assembly: A rate-distortion perspective," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Barcelona, Spain, Jul. 2016, pp. 1799–1803.

[32] E. Ukkonen, "Approximate string-matching with q-grams and maximal matches," *Theor. Comput. Sci.*, vol. 92, no. 1, pp. 191–211, Jan. 1992.

[33] S. M. H. T. Yazdi, Y. Yuan, J. Ma, H. Zhao, and O. Milenkovic, "A rewritable, random-access DNA-based storage system," *Sci. Rep.*, vol. 5, no. 1, pp. 1–10, Nov. 2015.

Sagi Marcovich (Student Member, IEEE) received the B.Sc. degree in computer science from the Technion—Israel Institute of Technology, Haifa, Israel, in 2016, where he is currently pursuing the Ph.D. degree with the Department of Computer Science. His research interests include algorithms, information theory, and coding theory with applications to DNA-based storage.

Eitan Yaakobi (Senior Member, IEEE) received the B.A. degree 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. 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 in 2010 and 2011.