Correcting Bursty and Localized Deletions Using Guess & Check Codes

Serge Kas Hanna, Salim El Rouayheb
ECE Department, Rutgers University
serge.k.hanna@rutgers.edu, salim.elrouayheb@rutgers.edu

***Abstract*—We consider the problem of constructing binary codes for correcting deletions that are localized within a certain part of the codeword that is unknown a priori. The model that we study is when $\delta \leq w$ deletions occur in a window of size at most w bits. These δ deletions are not necessarily consecutive, but are restricted to a window of size w. The localized deletions model is a generalization of the bursty model, where all the deleted bits are consecutive. In this work we propose new explicit codes, based on the family of Guess & Check codes [1,2], that can correct, with high probability, $\delta \leq w$ deletions that are localized within a window of size at most $w = \mathcal{O}(\log k)$, where k is the length of the information message. These codes have deterministic polynomial time encoding and decoding schemes. The redundancy of these codes is $c \log k + w + 1$, where c is a constant representing a code parameter.**

I. Introduction

Deletion and insertion errors are experienced in various communication and storage systems. These errors are often associated with various forms of loss of synchronization [3]–[6]. In many applications, the deletion and insertion errors tend to occur in bursts (consecutive errors), or are localized within certain parts of the information. This happens for example when synchronization is lost for a certain period of time, resulting in a series of errors within that time frame. Another example is in file synchronization (e.g. Dropbox), where large files are often edited by deleting and inserting characters in a relatively small part of the text (such as editing a paragraph), resulting in errors that are localized within that part of the file.

In this paper, we focus on deletion errors. Deletions were first explored in the 1960s [7]–[9]. In 1966, Levenshtein [8] showed that the codes constructed by Varshamov and Tenengolts (VT codes) [7] are capable of correcting a single deletion. Also in [8], Levenshtein derived non-constructive asymptotic bounds on the redundancy needed to correct deletions. The bounds showed that the redundancy needed to correct δ bit deletions in a codeword of length n bits is asymptotically $c\delta \log n$, for some constant $c > 0$. Moreover, for the case of a burst of exactly δ consecutive deletions, Levenshtein showed that at least $\log n + \delta - 1$ redundant bits are required. Levenshtein's bounds were later improved by Cullina and Kiyavash [10] for the general case of δ unrestricted deletions. Schoeny *et al.* [11] applied similar techniques as in [10] to derive a non-asymptotic lower bound for the case of a burst of δ deletions. The non-asymptotic bound derived in [11] matched Levenshtein's asymptotic bound.

Several previous works studied the general problem of constructing binary codes that correct multiple unrestricted deletions ($\delta > 1$) [1,2,12]–[14]. Intuitively, correcting a burst of two deletions is an easier problem compared to correcting two deletions that are not necessarily consecutive. This idea is also reflected through Levenshtein's bounds on the redundancy, which indicate that less redundancy is required when the deletions occur in a burst.

Levenshtein in [15] constructed asymptotically optimal codes that can correct a burst of at most two deletions. Cheng *et al.* [16] provided three constructions of codes that can correct a burst of *exactly* $\delta > 2$ deletions. The lowest redundancy achieved by the codes in [16] is $\delta(\log(n/\delta + 1))$. The fact that the number of deletions in the burst is exactly δ is a crucial factor in the code constructions in [16]. This is because the high-level idea of the construction in [16] is to arrange the codeword in a matrix of size $(n/\delta) \times \delta$ such that each row of this matrix belongs to a VT code. In this case, a single burst of δ deletions deletes exactly one bit in each row of the matrix. The received string is then arranged in a matrix of size $(n/\delta) \times (\delta - 1)$, and each row is then corrected using VT codes. Note that the constructions in [16] do not cover the case of a burst of *at most* δ deletions. Schoeny *et al.* [11] prove the *existence* of codes that can correct a burst of exactly δ deletions and have at most $\delta \log(n) + (\delta - 1) \log(\log n) + \delta - 1$ redundancy, for sufficiently large n. Schoeny *et al.* [11] also prove the existence of codes that can correct a burst of at most δ deletions, their results for this model improve on a previous result by Bours in [17]. In this paper, we focus on the more general model, where $\delta \leq w$ deletions are localized within a window of size at most w bits, but are not necessarily consecutive. To the best of our knowledge, the only previous work on this model is the one by Schoeny *et al.* in [11], where the authors prove the existence of codes that can correct localized deletions for the particular cases of $w = 3$ and $w = 4$.

In this paper, we extend our work in [1] where we proposed the Guess & Check codes for the general problem of correcting $\delta \geq 1$ unrestricted deletions. We exploit the localized nature of the deletions to modify the schemes in [1] and obtain explicit codes that can correct $\delta \leq w$ localized deletions, with $w = \mathcal{O}(\log k)$. Similar to [1], the codes that we propose

This paper will appear in the proceedings of the 55th Annual Allerton Conference on Communication, Control, and Computing.
This work was supported in parts by NSF Grant CCF 15-26875.
Part of this work was done while the authors were with the ECE department at the Illinois Institute of Technology.

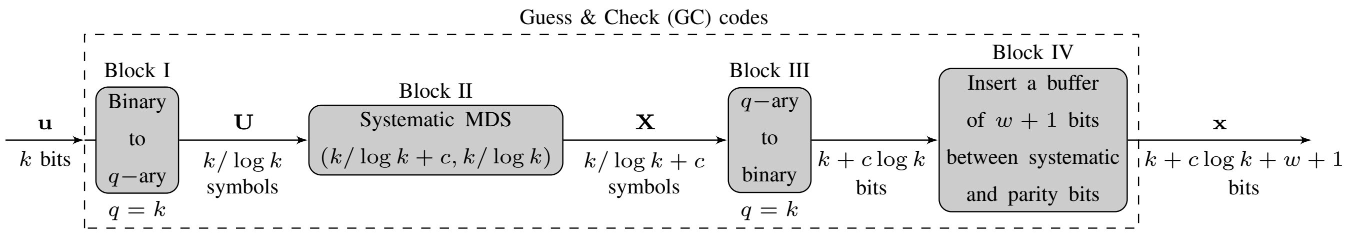

Fig. 1: Encoding block diagram of GC codes for correcting $\delta \leq w$ deletions that are localized within a window of size at most w bits. Block I: The binary message of length k bits is chunked into adjacent blocks of length $\log k$ bits each, and each block is mapped to its corresponding symbol in $GF(q)$ where $q = 2^{\log k} = k$. Block II: The resulting string is coded using a systematic $(k/\log k + c, k/\log k)$ $q-$ary MDS code where c is the number of parity symbols and $q = k > k/\log k + c$. Block III: The symbols in $GF(q)$ are mapped to their binary representations. Block IV: A buffer of w zeros followed by a single one is inserted between the systematic and the parity bits.

in this paper have an asymptotically vanishing probability of decoding failure[1], assuming uniform i.i.d. message. We also assume that the positions of the deletions are independent of the information message. Note that the previous works on related models (e.g., [11,16,17]) focus on constructing zero-error codes, but in most cases it remains open whether these zero-error based constructions can yield explicit polynomial time encoding and decoding schemes for the problem.

Contributions: (i) We propose new explicit codes, based on the family of Guess & Check codes, that can correct, with high probability, $\delta \leq w$ deletions that are localized within a single window of size $w = \mathcal{O}(\log k)$ bits; (ii) These codes have near-linear time encoding, and cubic time decoding; (iii) The redundancy of these codes is $c\log k + w + 1$ bits, i.e., logarithmic in the length of the message k, and linear in the length of the window w; (iv) We provide numerical simulations on the performance of these codes, and compare the simulation results to our theoretical bounds.

II. Preliminaries

In this paper, we consider the following models of deletions:

1) Burst of fixed length: the deletions occur in a burst of size $\delta \geq 1$, i.e., exactly δ consecutive bits are deleted.
2) Burst of variable length: at most $\delta \geq 1$ consecutive bits are deleted.
3) Localized deletions: $\delta \leq w$ deletions are localized within a single window of size at most δ bits. In this model, the δ deletions do not necessarily occur in a burst, i.e., are not necessarily consecutive.

Example 1 (Localized Deletions)**.** *Let $w = 6$ and $\delta = 4$. Let $\mathbf{x}$ be the transmitted string and $\mathbf{y}$ be the received string. The bits in red represent $\delta = 4$ localized deletions:*

$$\mathbf{x} = 1\ 0\ \overbrace{0\ 1\ 0\ 1\ 0\ 1}^{\text{window}}\ 0\ 1\ 0\ 0\ 1\ 1\ 1\ 0,$$
$$\mathbf{y} = 1\ 0\ 0\ 0\ 0\ 1\ 0\ 0\ 1\ 1\ 1\ 0.$$

Note that the two bursty models of deletions correspond to special cases of the localized deletions model. Hence, a code that can correct localized deletions can also correct a burst of fixed or variable length.

4) Unrestricted deletions: at most $\delta \geq 1$ deletions can occur anywhere in the string.

Furthermore, we assume that the input message is uniform i.i.d. and that the locations of the deletions are independent of the message. We denote by k and n the lengths in bits of the message and codeword, respectively. We assume without loss of generality that k is a power of 2. The encoding block diagram for correcting localized deletions is shown in Fig. 1. We denote binary and $q-$ary vectors by lower and upper case bold letters respectively, and random variables by calligraphic letters. All logarithms in this paper are of base 2. We denote by F the event that corresponds to a decoding failure, i.e., when the decoder cannot make a correct decision and outputs a "failure to decode" error message.

III. Main Results

Consider the encoding block diagram of GC codes shown in Fig. 1. The input message $\mathbf{u}$ is uniform i.i.d., i.e., the k bits of $\mathbf{u}$ are i.i.d. Bernoulli(1/2).

Theorem 1. *Guess & Check (GC) codes can correct in polynomial time $\delta \leq w$ deletions that are localized within a single window of size at most w bits, where $m\log k + 1 \leq w \leq (m+1)\log k + 1$ for some constant integer $m \geq 0$. Let $c > m+2$ be a constant integer. The code has the following properties:*

1) Redundancy: $n - k = c\log k + w + 1$ bits.
2) Encoding complexity is $\mathcal{O}(k\log k)$, and decoding complexity is $\mathcal{O}\left(k^3/\log k\right)$.
3) Probability of decoding failure:

$$Pr(F) \leq \frac{k^{m+4}}{k^c \log k} - (m+2)\frac{k^{m+3}}{k^c}. \qquad (1)$$

The code properties in Theorem 1 show that: (i) the redundancy is logarithmic in the length of the message k, and linear in the length of the window w; (ii) the encoding complexity is near-linear, and the decoding complexity is cubic; (iii) the probability of decoding failure vanishes asymptotically in k if $c \geq m+4$. Moreover, the probability of decoding failure decreases exponentially in c for a fixed k.

[1]The term *decoding failure* means that the decoder cannot make a correct decision and outputs a "failure to decode" error message.

IV. Examples

In this section, we provide encoding and decoding examples of GC codes for correcting $\delta \leq w$ deletions that are localized within a single window of size $w = \log k$ bits.

Example 2 (Encoding)**.** *Consider a message* $\mathbf{u}$ *of length* $k = 16$ *given by* $\mathbf{u} = 1100101001111000$. $\mathbf{u}$ *is encoded by following the different encoding blocks illustrated in Fig. 1.*
1) Binary to $q-$ary (Block I, Fig. 1). *The message* $\mathbf{u}$ *is chunked into adjacent blocks of length* $\log k = 4$ *bits each,*

$$\mathbf{u} = \underbrace{\overset{\text{block 1}}{\overbrace{1\,1\,0\,0}}}_{\alpha^6}\ \underbrace{\overset{\text{block 2}}{\overbrace{1\,0\,1\,0}}}_{\alpha^9}\ \underbrace{\overset{\text{block 3}}{\overbrace{0\,1\,1\,1}}}_{\alpha^{10}}\ \underbrace{\overset{\text{block 4}}{\overbrace{1\,0\,0\,0}}}_{\alpha^3}.$$

Each block is then mapped to its corresponding symbol in $GF(q)$, $q = k = 2^4 = 16$. *This results in a string* $\mathbf{U}$ *which consists of* $k/\log k = 4$ *symbols in* $GF(16)$. *The extension field used here has a primitive element* α, *with* $\alpha^4 = \alpha + 1$. *Hence, we obtain* $\mathbf{U} = (\alpha^6, \alpha^9, \alpha^{10}, \alpha^3) \in GF(16)^4$.
2) Systematic MDS code (Block II, Fig. 1). $\mathbf{U}$ *is then coded using a systematic* $(k/\log k + c, k/\log k) = (7, 4)$ *MDS code over* $GF(16)$, *with* $c = 3$. *The encoded string is denoted by* $\mathbf{X} \in GF(16)^3$ *and is given by multiplying* $\mathbf{U}$ *by the following code generator matrix*[2]

$$\mathbf{X} = \left(\alpha^6, \alpha^9, \alpha^{10}, \alpha^3\right) \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & \alpha & \alpha^2 \\ 0 & 0 & 1 & 0 & 1 & \alpha^2 & \alpha^4 \\ 0 & 0 & 0 & 1 & 1 & \alpha^3 & \alpha^6 \end{pmatrix},$$
$$= (\alpha^6, \alpha^9, \alpha^{10}, \alpha^3, \alpha^{14}, \alpha^3, \alpha).$$

3) $Q-$ary to binary (Block III, Fig. 1). *The binary representation of* $\mathbf{X}$, *of length* $n = k + 3\log k = 28$ *bits, is* 1100 1010 0111 1000 1001 1000 0001.
4) Adding a buffer of $w + 1$ bits (Block IV, Fig. 1). *A buffer of* $w = \log k = 4$ *zeros followed by single one is inserted between the systematic and parity bits. The binary codeword to be transmitted is of length 33 bits and is given by*

$$\mathbf{x} = 1100\ 1010\ 0111\ 1000\ \overset{\text{buffer}}{\overbrace{\mathbf{00001}}}\ 1001\ 1000\ 0001.$$

Now we explain the high-level idea of decoding. Since $\delta \leq w$ and $w = \log k$, then the δ deletions can affect at most two adjacent blocks in $\mathbf{x}$. The goal of the decoder is to recover the systematic part of $\mathbf{x}$, so it considers following guesses: 1) blocks 1 and 2 are affected by the deletions; 2) blocks 2 and 3 are affected; 3) blocks 3 and 4 are affected. For each of these guesses the decoder: (i) chunks the received sequence based on the guess on the location of the two affected blocks; (ii) considers the two affected blocks erased and decodes them by using the first two MDS parity symbols; (iii) checks whether the the decoded string is consistent with the third MDS parity and with the received sequence.

[2]The generator matrix used here is obtained by concatenating an identity matrix with a Vandermonde matrix. This approach does not always yield MDS codes, for the general case we use Cauchy matrices instead of Vandermonde matrices.

Example 3 (Decoding)**.** *Suppose that the* 7th, 9th *and* 10th *bit of* $\mathbf{x}$ *are deleted. Hence, the decoder receives the following 30 bit string* $\mathbf{y}$,

$$\mathbf{y} = 1100100111000\mathbf{0001}100110000001.$$

Note that the window size w *is known at the decoder, and the number of deletions* δ *can be determined by the difference between* n *(code parameter) and the length of the received string* $\mathbf{y}$. *In this example, we have* $w = \log k = 4$ *and* $\delta = 3$. *Moreover, the localized* $\delta \leq w$ *deletions cannot affect both the systematic and the parity bits simultaneously, since these two are separated by a buffer of size* $w + 1$ *bits. Therefore, we consider the following two scenarios: (i) If the deletions affected the parity bits, then the decoder directly outputs the systematic bits and decoding will be over*[3]*; (ii) If the deletions affected the systematic bits, then the decoder goes over the guesses as explained previously.*

The buffer is what allows the decoder to determine which of the two previous scenarios to consider. The decoder observes the $(k + w - \delta + 1)^{th} = 18^{th}$ *bit in* $\mathbf{y}$,

$$1100100111000\mathbf{000}\underline{\mathbf{1}}100110000001.$$

Based on the value of the observed bit, the decoder can determine whether the deletions affected the systematic bits or not. The previous operation can be done with zero-error, we explain it in more detail in Section V. In this example, the fact that the observed bit is a one indicates that the one in the buffer has shifted δ *positions to the left. Hence, the decoder considers that the deletions have affected the systematic bits, and thus proceeds with making its guesses as we explain next. Henceforth, we remove the buffer from the string and consider only the systematic and parity bits.*

The decoder goes through all the possible $k/\log k - 1 = 3$ *cases (guesses), where in each case* i, $i = 1, \ldots, 3$, *the deletions are assumed to have affected blocks* i *and* $i+1$, *and* $\mathbf{y}$ *is chunked accordingly. Given this assumption, symbols* i *and* $i + 1$ *are considered erased and erasure decoding is applied over* $GF(16)$ *to recover these two symbols. Without loss of generality, we assume that the first two parities* $p_1 = \alpha^{14}$ *and* $p_2 = \alpha^3$ *are used for decoding the two erasures. The decoded* $q-$*ary string in case* i *is denoted by* $\mathbf{Y_i} \in GF(16)^4$, *and its binary representation is denoted by* $\mathbf{y_i} \in GF(2)^{16}$. *The three cases are shown below:*
Case 1: *The deletions are assumed to have affected blocks 1 and 2. Hence,* $\mathbf{y}$ *is chunked as follows*

$$\underbrace{1\,1\,0\,0\,1}_{\mathcal{E}}\ \underbrace{0\,0\,1\,1}_{\alpha^4}\ \underbrace{1\,0\,0\,0}_{\alpha^3}\ \underbrace{1\,0\,0\,1}_{\alpha^{14}}\ \underbrace{1\,0\,0\,0}_{\alpha^3}\ \underbrace{0\,0\,0\,1}_{1},$$

where $\mathcal{E}$ *denotes the bits corresponding to symbols 1 and 2 that are considered to be erased. Applying erasure decoding over* $GF(16)$, *the recovered values of symbols 1 and 2 are* α^2 *and* α^5, *respectively. Hence, the decoded* $q-$*ary string* $\mathbf{Y_1} \in GF(16)^4$ *is*

$$\mathbf{Y_1} = (\alpha^2, \alpha^5, \alpha^4, \alpha^3).$$

[3]The information is in the systematic bits, hence the decoder does need to recover parity bits.

Its equivalent in binary $\mathbf{y_1} \in GF(2)^{16}$ is

$$\mathbf{y_1} = \underbrace{0\,1\,0\,0}_{\alpha^2}\ \underbrace{0\,1\,1\,0}_{\alpha^5}\ \underbrace{0\,0\,1\,1}_{\alpha^4}\ \underbrace{1\,0\,0\,0}_{\alpha^3}.$$

Notice that the concatenated binary representation of the two decoded erasures (01000110)*, is not a supersequence of the sub-block* (11001)*, which was denoted by $\mathcal{E}$. Hence, the decoder can immediately determine that the assumption in this case is wrong, i.e., the deletions did not affect blocks 1 and 2. Throughout the paper we refer to such cases as* impossible *cases. Another way for the decoder to check whether this case is possible is to test if $\mathbf{Y_1}$ is consistent with the third parity $p_3 = 1$. However, the computed parity is*

$$\left(\alpha^2, \alpha^5, \alpha^4, \alpha^3\right)\left(1, \alpha^2, \alpha^4, \alpha^6\right)^T = 0 \neq 1.$$

Therefore, this is an additional reason that proves that case 1 is impossible.
Case 2: *The deletions are assumed to have affected blocks 2 and 3, so the sequence is chunked as follows*

$$\underbrace{1\,1\,0\,0}_{\alpha^6}\ \underbrace{1\,0\,0\,1\,1}_{\mathcal{E}}\ \underbrace{1\,0\,0\,0}_{\alpha^3}\ \underbrace{1\,0\,0\,1}_{\alpha^{14}}\ \underbrace{1\,0\,0\,0}_{\alpha^3}\ \underbrace{0\,0\,0\,1}_{1}.$$

Applying erasure decoding, the recovered values of symbols 2 and 3 are α^9 and α^{10}, respectively. The decoded binary string is

$$\mathbf{y_2} = \underbrace{1\,1\,0\,0}_{\alpha^6}\ \underbrace{1\,0\,1\,0}_{\alpha^9}\ \underbrace{0\,1\,1\,1}_{\alpha^{10}}\ \underbrace{1\,0\,0\,0}_{\alpha^3}\ .$$

In this case, the concatenated binary representation of the two decoded erasures (10100111) *is a supersequence of the sub-block* (10011)*. Moreover, it is easy to verify that the decoded string is consistent with the third parity $p_3 = 1$. Therefore, we say that case 2 is* possible.
Case 3: *The deletions are assumed to have affected blocks 3 and 4, so the sequence is chunked as follows*

$$\underbrace{1\,1\,0\,0}_{\alpha^6}\ \underbrace{1\,0\,0\,1}_{\alpha^{14}}\ \underbrace{1\,1\,0\,0\,0}_{\mathcal{E}}\ \underbrace{1\,0\,0\,1}_{\alpha^{14}}\ \underbrace{1\,0\,0\,0}_{\alpha^3}\ \underbrace{0\,0\,0\,1}_{1}.$$

The decoded binary string is

$$\mathbf{y_3} = \underbrace{1\,1\,0\,0}_{\alpha^6}\ \underbrace{1\,0\,0\,1}_{\alpha^{14}}\ \underbrace{1\,1\,0\,0}_{\alpha^6}\ \underbrace{0\,0\,0\,0}_{0}\ .$$

In this case, the concatenated binary representation of the two decoded erasures (11000000) *is a supersequence of the sub-block* (11000)*. However, it is easy to verify that the decoded string is not consistent with $p_3 = 1$. Therefore, case 3 is* impossible.

After going through all the cases, case 2 stands alone as the only possible *case. So the decoder declares successful decoding and outputs $\mathbf{y_2}$ ($\mathbf{y_2} = \mathbf{u}$).*

Remark 1. *Sometimes the decoder may find more than one* possible *case resulting in different decoded strings. In that situation, the decoder cannot know which of the cases is the correct one, so it declares a decoding failure. Although a decoding failure may occur, Theorem 1 indicates that its probability vanishes as length of the message k goes to infinity.*

V. Encoding and Decoding for Localized Deletions

As previously mentioned, our schemes for correcting localized deletions extend from the encoding and decoding schemes of Guess & Check (GC) codes in [1], which are designed for correcting δ unrestricted deletions. In this section, we give an overview of the encoding and decoding schemes in [1], and explain how we exploit the localized nature of the deletions to modify these schemes and obtain codes having the properties shown in Theorem 1. We first start by restating the main result in [1].

Theorem 2. *([1]) Guess & Check (GC) codes can correct in polynomial time up to a constant number of δ deletions. Let $c > \delta$ be a constant integer. The code has the following properties:*

1) Redundancy: $n - k = c(\delta + 1)\log k$ bits.

2) Encoding complexity is $\mathcal{O}(k \log k)$, and decoding complexity is $\mathcal{O}\left(k^{\delta+2}/\log^{\delta} k\right)$.

3) Probability of decoding failure:

$$Pr(F) = \mathcal{O}\left(\frac{1}{k^{c-2\delta}\log^{\delta} k}\right). \tag{2}$$

Next, we explain the encoding and decoding using GC codes for the following two models: (1) Unrestricted deletions, i.e., up to δ deletions that are not necessarily localized; (2) Localized deletions.

A. Encoding using GC codes

For both models mentioned above, the first three encoding blocks are the same as the ones shown in Fig. 1. As for the last encoding block (Block IV):

1) Unrestricted deletions: The parity bits are encoded using a $(\delta + 1)$ repetition code[4]. The repetition code protects the parities against any deletions, and allows them to be recovered at the decoder.

2) Localized deletions: The systematic and parity bits are separated by a buffer of size $w + 1$ bits, which consists of w zeros followed by a single one. In the upcoming decoding section, we explain how the decoder uses this buffer to detect whether the deletions affected the systematic bits or not.

B. Decoding using GC codes

1) Unrestricted deletions [1]: The approach presented in [1] for decoding up to δ deletions is the following:
(a) Decoding the parity bits: the decoder recovers the parity bits which are protected by a $(\delta + 1)$ repetition code.
(b) The guessing part: the number of possible ways to distribute the δ deletions among the $k/\log k$ blocks is $t = \binom{k/\log k + \delta - 1}{\delta}$. These possibilities are indexed by $i, i = 1, \ldots, t$, and each possibility is referred to by case i.
(c) The checking part: for each case i, $i = 1, \ldots, t$, the decoder (i) chunks the sequence based on the corresponding assumption on the locations of the δ deletions; (ii) considers

[4]Each parity bit is repeated $(\delta + 1)$ times.

the affected blocks erased and maps the remaining blocks to their corresponding symbols in $GF(q)$; (iii) decodes the erasures using the first δ parity symbols; (iv) checks whether the case is *possible* or not by testing if the decoded string is consistent with the received string and with the last $c-\delta$ parity symbols. The criteria used to check if a case is possible or not are given in Definition 1.

Definition 1. *For δ deletions, a case i, $i=1,\dots,t$, is said to be* possible *if it satisfies the following two criteria simultaneously. Criterion 1: the decoded $q-$ary string in case i satisfies the last $c-\delta$ parities simultaneously. Criterion 2: the binary representations of all the decoded erasures are supersequences of their corresponding sub-blocks.*

2) Localized Deletions: Consider the case of decoding $\delta \leq w$ deletions that are localized within a single window of size at most w bits. Notice that if $m\log k+1 \leq w \leq (m+1)\log k+1$, for some constant integer $m \geq 0$, then the δ deletions can affect at most $m+2$ adjacent blocks in the codeword. Therefore, in terms of the GC code construction, correcting the δ deletions corresponds to decoding at most $m+2$ block erasures. Hence, the localized nature of the deletions enables the following simplifications to the scheme: (i) The total number of cases to be checked by the decoder is reduced to $k/\log k-m-1$ since at most $m+2$ adjacent blocks (out of $k/\log k$ blocks) can be affected by the deletions; (ii) Instead of protecting the parity bits by a $(\delta+1)$ repetition code, they are separated from the systematic bits by a buffer of size $w+1$ bits, composed of w zeros followed by a single one.

Now we explain the decoding steps. Note that the $\delta \leq w$ localized deletions cannot affect the systematic and the parity bits simultaneously since these two are separated by a buffer of size $w+1$ bits. The decoder uses this buffer to detect whether the deletions have affected the systematic bits or not. The buffer is composed of w zeros followed by a single one, and its position ranges from the $(k+1)^{th}$ bit to the $(k+w+1)^{th}$ bit of the transmitted string. Let $\mathbf{y}_\lambda$ be the bit in position λ in the received string, where $\lambda \triangleq k+w-\delta+1$. The decoder observes $\mathbf{y}_\lambda$ [5]: (1) If $\mathbf{y}_\lambda=1$, then this means that the one in the buffer has shifted δ positions to the left because of the deletions, i.e., all the deletions occurred to the left of the one in the buffer. In this case, the decoder considers that the deletions affected the systematic bits, and therefore proceeds to the guessing and checking part. The same steps are applied as in the case of δ deletions, while considering a total of $k/\log k-m-1$ cases, each corresponding to $m+2$ block erasures. In each case, the last $c\log k$ bits of the received string (parities) are used to decode the first $k-\delta$ bits. (2) If $\mathbf{y}_\lambda=0$, then this indicates that the δ deletions occurred to the right of the first zero in the buffer, i.e., the systematic bits were unaffected. In this case, the decoder simply outputs the first k bits of the received string.

[5]The decoder knows the values of k, n, and w (code parameters), and can determine the value of δ by calculating the difference between n and the length of the received string.

VI. Simulation Results

We tested the performance of GC codes for correcting $\delta=\log k-1$ deletions that are localized within a single window of size $w=\log k$ bits. We performed numerical simulations for messages of length $k=256, 512$ and 1024 bits. We also compared the obtained probability of decoding failure to the upper bound in Theorem 1.

Config.	c					
	3		4		5	
k	R	$Pr(F)$	R	$Pr(F)$	R	$Pr(F)$
256	0.86	$6.1e^{-2}$	0.84	$3.5e^{-4}$	0.82	0
512	0.92	$5.8e^{-2}$	0.90	$1.1e^{-4}$	0.89	0
1024	0.96	$5.4e^{-2}$	0.95	$9.0e^{-5}$	0.94	0

TABLE I: The table shows the code rate $R=k/n$ and the probability of decoding failure $Pr(F)$ of GC codes for different message lengths k and number of MDS parity symbols c. The simulations are for $\delta=\log k-1$ deletions that are localized within a window of size $w=\log k$ bits. The values of R are rounded to the nearest 10^{-2}. The results of $Pr(F)$ are averaged over 10^5 runs of simulations. In each run, a message $\mathbf{u}$ chosen uniformly at random is encoded into the codeword $\mathbf{x}$. The positions of the window and the deletions in $\mathbf{x}$ are correspondingly chosen uniformly at random. The resulting string is then decoded.

To guarantee an asymptotically vanishing probability of decoding failure, the upper bound in Theorem 1 requires that $c \geq 4$. In fact, the bound shows that the probability of decoding failure decreases logarithmically in k for $c=4$, whereas for $c>4$ the probability decreases polynomially. Therefore, we make a distinction between three regimes, (i) $c=3$: Here, the theoretical upper bound is trivial. Whereas the simulation results in Table I show that the probability of decoding failure is of the order of 10^{-2}. This indicates that GC codes can decode correctly in this regime, although not reflected in the upper bound; (ii) $c=4$: The upper bound is of the order of 10^{-1} for $k=1024$, whereas the probability of decoding failure recorded in the simulations is of the order of 10^{-5}. (iii) $c>4$: The upper bound is of the order of 10^{-4} for $k=1024$ and $c=5$, and of the order of 10^{-7} for $k=1024$ and $c=6$. In the simulations no decoding failure was detected within 10^5 runs for $c=5$ and $c=6$. In general, the simulations show that GC codes perform better than what the upper bound indicates. These simulations were performed on a personal computer. The average decoding time is about $40\ \mu s$ for $k=256$, $67\ \mu s$ for $k=512$, and $117\ \mu s$ for $k=1024$.

VII. Proof of Theorem 1

A. Redundancy

The redundancy follows from the construction in Fig. 1. The number of redundant bits is $c\log k+w+1$ (parities + buffer).

B. Complexity

The encoding complexity is $\mathcal{O}(k\log k)$, same as the case of unrestricted deletions [1]. The dominant factor in the decoding complexity is the part where the decoder goes over all the possible cases and applies erasure decoding for each case [1]. Hence, the order of decoding complexity is given by the total number of cases multiplied by the complexity of erasure decoding

$$\mathcal{O}\left(\left(\frac{k}{\log k}-m-1\right)\times k^2\right)=\mathcal{O}\left(\frac{k^3}{\log k}\right).$$

C. Probability of Decoding Failure

The proof of the upper bound on the probability of decoding failure follows similar techniques as the ones used to prove the main result in [1] (Theorem 2). The analysis requires some modifications to deal with the case of localized deletions. For the sake of completeness, we go over all the steps in detail.

The probability of decoding failure is computed over all possible $k-$bit messages. Recall that the message $\mathbf{u}$ is uniform i.i.d., i.e., the bits of $\mathbf{u}$ are i.i.d. Bernoulli$(1/2)$. The message $\mathbf{u}$ is encoded as shown in Fig. 1. Consider $\delta \leq w$ deletions that are localized within a window of size at most w bits, where $m\log k+1 \leq w \leq (m+1)\log k+1$, for some constant integer $m \geq 0$. In this case, the δ deletions can affect at most $m+2$ adjacent blocks in the codeword. Therefore, the decoder goes through $k/\log k-m-1$ cases, where in each case it assumes that certain $m+2$ adjacent blocks were affected by the deletions. Let $\boldsymbol{\mathcal{Y}_i}$ be the random variable representing the $q-$ary string decoded in case i, $i=1,2,\dots,k/\log k-m-1$. Let $\mathbf{Y}\in GF(q)^{k/\log k}$ be a realization of the random variable $\boldsymbol{\mathcal{Y}_i}$. We denote by $\mathcal{P}_r \in GF(q), r=1,2,\dots,c$, the random variable representing the r^{th} MDS parity symbol. Also, let $\mathbf{G_r}\in GF(q)^{k/\log k}$ be the MDS encoding vector responsible for generating $\mathcal{P}_r$. Consider $c > m+2$ arbitrary MDS parities $p_1,\dots,p_c$, for which we define the following sets. For $r=1,\dots,c$,

$$\mathrm{A_r} \triangleq \{\mathbf{Y}\in GF(q)^{k/\log k} |\ \mathbf{G_r^T}\mathbf{Y}=p_r\},$$
$$\mathrm{A_1^r} \triangleq \mathrm{A_1}\cap\mathrm{A_2}\cap\ldots\cap\mathrm{A_r}.$$

$\mathrm{A_r}$ and $\mathrm{A_1^r}$ are affine subspaces of dimensions $k/\log k-1$ and $k/\log k-r$, respectively. Therefore,

$$|\mathrm{A_r}| = q^{\frac{k}{\log k}-1} \text{ and } |\mathrm{A_1^r}| = q^{\frac{k}{\log k}-r}. \tag{3}$$

$\boldsymbol{\mathcal{Y}_i}$ is decoded based on the first $m+2$ parities, therefore, $\boldsymbol{\mathcal{Y}_i}\in\mathrm{A_1^{m+2}}$. Note that $\boldsymbol{\mathcal{Y}_i}$ is not necessarily uniformly distributed over $\mathrm{A_1^{m+2}}$. The next claim gives an upper bound on the probability mass function of $\boldsymbol{\mathcal{Y}_i}$ for given arbitrary parities.

Claim 1. *For any case i, $i=1,2,\dots,k/\log k-m-1$,*

$$Pr\left(\boldsymbol{\mathcal{Y}_i}=\mathbf{Y}|\mathcal{P}_1=p_1,\dots,\mathcal{P}_{m+2}=p_{m+2}\right)\leq \frac{k}{q^{\frac{k}{\log k}-m-2}}.$$

We assume Claim 1 is true for now and prove it shortly. Next, we use this claim to prove the upper bound on the probability of decoding failure in (1).

In the general decoding scheme, there are two criteria which determine whether a case is *possible* or not (Definition 1). Here, we upper bound $Pr(F)$ by taking into account Criterion 1 only. Based on Criterion 1, if a case i is possible, then $\boldsymbol{\mathcal{Y}_i}$ satisfies all the c MDS parities simultaneously, i.e., $\boldsymbol{\mathcal{Y}_i}\in\mathrm{A_1^c}$. Without loss of generality, we assume case 1 is the correct case. A decoding failure is declared if there exists a *possible* case j, $j=2,\dots,k/\log k-m-1$, that leads to a decoded string different than that of case 1. Namely, $\boldsymbol{\mathcal{Y}_j}\in\mathrm{A_1^c}$ and $\boldsymbol{\mathcal{Y}_j}\neq\boldsymbol{\mathcal{Y}_1}$. Let $t=k/\log k-m-1$ and $\mathbf{p_1^{m+2}}=(p_1,\dots,p_{m+2})$, we have

$$Pr\left(F|\mathbf{p_1^{m+2}}\right)\leq Pr\left(\bigcup_{j=2}^{t}\{\boldsymbol{\mathcal{Y}_j}\in\mathrm{A_1^c},\boldsymbol{\mathcal{Y}_j}\neq\boldsymbol{\mathcal{Y}_1}\}\middle|\mathbf{p_1^{m+2}}\right) \tag{4}$$

$$\leq \sum_{j=2}^{t} Pr\left(\boldsymbol{\mathcal{Y}_j}\in\mathrm{A_1^c},\boldsymbol{\mathcal{Y}_j}\neq\boldsymbol{\mathcal{Y}_1}|\mathbf{p_1^{m+2}}\right) \tag{5}$$

$$\leq \sum_{j=2}^{t} Pr\left(\boldsymbol{\mathcal{Y}_j}\in\mathrm{A_1^c}|\mathbf{p_1^{m+2}}\right) \tag{6}$$

$$= \sum_{j=2}^{t}\sum_{\mathbf{Y}\in\mathrm{A_1^c}} Pr\left(\boldsymbol{\mathcal{Y}_j}=\mathbf{Y}|\mathbf{p_1^{m+2}}\right) \tag{7}$$

$$\leq \sum_{j=2}^{t}\sum_{\mathbf{Y}\in\mathrm{A_1^c}} \frac{k}{q^{\frac{k}{\log k}-m-2}} \tag{8}$$

$$= \sum_{j=2}^{t} |\mathrm{A_1^c}|\,\frac{k}{q^{\frac{k}{\log k}-m-2}} \tag{9}$$

$$= \sum_{j=2}^{t} k\ \frac{q^{\frac{k}{\log k}-c}}{q^{\frac{k}{\log k}-m-2}} \tag{10}$$

$$= k\,(t-1)\,\frac{q^{m+2}}{q^c} \tag{11}$$

$$= \frac{k^{m+3}}{k^c}\left(\frac{k}{\log k}-m-2\right) \tag{12}$$

$$= \frac{k^{m+4}}{k^c\log k}-(m+2)\frac{k^{m+3}}{k^c}. \tag{13}$$

(5) follows from applying the union bound. (6) follows from the fact that $Pr\left(\boldsymbol{\mathcal{Y}_j}\neq\boldsymbol{\mathcal{Y}_1}|\boldsymbol{\mathcal{Y}_j}\in\mathrm{A_1^c},\mathbf{p_1^{m+2}}\right)\leq 1$. (8) follows from Claim 1. (10) follows from (3). (12) follows from the fact that $q=k$ and $t=k/\log k-m-1$. Next, to complete the proof of (1), we use (13) and average over all values of $p_1,\dots,p_{m+2}$

$$Pr(F) = \sum_{\mathbf{p_1^{m+2}}\in GF(q^{m+2})} Pr\left(F|\mathbf{p_1^{m+2}}\right)Pr\left(\mathbf{p_1^{m+2}}\right)$$
$$\leq \frac{k^{m+4}}{k^c\log k}-(m+2)\frac{k^{m+3}}{k^c}.$$

D. Proof of Claim 1

Recall that $\boldsymbol{\mathcal{Y}_i}\in\mathrm{A_1^{m+2}}$ is the random variable representing the output of the decoder in case i. Claim 1 gives an upper bound on the probability mass function of $\boldsymbol{\mathcal{Y}_i}$ for any i and for given arbitrary parities $(p_1,\dots,p_{m+2})$. To find the bound in Claim 1, we focus on an arbitrary case i (i fixed) that assumes that the deletions have affected blocks $i,i+1,\dots,i+k/\log k-m-2$. We observe $\boldsymbol{\mathcal{Y}_i}$ for all possible input $k-$bit messages, for a fixed deletion window[6],

[6]The window of w bits, in which the δ deletions are localized, is fixed.

and given parities $(p_1, \dots, p_{m+2})$. Hence, the observed case, the deletion window, and parities are fixed, while the input message varies. In this setting, we determine the maximum number of different inputs that can generate the same output. We call this number γ. Once we obtain γ we can write

$$Pr\left(\boldsymbol{\mathcal{Y}_i} = \mathbf{Y} | W, p_1, ..., p_{m+2}\right) \leq \frac{\gamma}{\left|\mathrm{A}_1^{\mathrm{m+2}}\right|} = \frac{\gamma}{q^{\frac{k}{\log k} - m - 2}}, \tag{14}$$

where W is an arbitrary window of size w bits in which the δ deletions are localized. We will explain our approach for determining γ by going through an example for $k = 32$ that can be generalized for any k. We denote by $b_o \in GF(2)$, $o = 1, 2, \dots, k$, the bit of the message $\mathbf{u}$ in position o.

Example 4. *Let $k = 32$ and $\delta = w = \log k + 1 = 6$. Consider the binary message $\mathbf{u}$ given by*

$$\mathbf{u} = b_1\ b_2\ \dots\ b_{32}.$$

Its corresponding $q-$ary message $\mathbf{U}$ consists of 7 symbols (blocks) of length $\log k = 5$ bits each[7]*. The message $\mathbf{u}$ is encoded into a codeword $\mathbf{x}$ as shown in Fig. 1. We assume that the first parity is the sum of the systematic symbols and the encoding vector for the second parity is $(1, \alpha, \alpha^2, \alpha^3, \alpha^4, \alpha^5, \alpha^6)$* [8]*. Suppose that the $\delta = w = 6$ deleted bits in $\mathbf{x}$ are b_3 up to b_8. Note that since $w = \log k + 1$, then $m = 0$, and this means that the deletions can affect at most $m + 2 = 2$ blocks in $\mathbf{x}$. Suppose that the case considered by the decoder is the one that assumes that the deletions affected the 3^{rd} and 4^{th} block (wrong case). The decoder chunks the codeword accordingly, and symbols 3 and 4 are considered to be erased. The rest of the $q-$ary symbols are given by*

$$\begin{aligned}
S_1 &= \alpha^4 b_1 + \alpha^3 b_2 + \alpha^2 b_9 + \alpha b_{10} + b_{11}, \\
S_2 &= \alpha^4 b_{12} + \alpha^3 b_{13} + \alpha^2 b_{14} + \alpha b_{15} + b_{16}, \\
S_5 &= \alpha^4 b_{21} + \alpha^3 b_{22} + \alpha^2 b_{23} + \alpha b_{24} + b_{25}, \\
S_6 &= \alpha^4 b_{26} + \alpha^3 b_{27} + \alpha^2 b_{28} + \alpha b_{29} + b_{30}, \\
S_7 &= \alpha^4 b_{31} + \alpha^3 b_{32}.
\end{aligned}$$

Notice that S_1, S_2, S_5, S_6 and S_7 are directly determined by the bits of $\mathbf{u}$ which are chunked at their corresponding positions. Hence, in order to obtain the same output, the bits of the inputs corresponding to these symbols cannot be different. For instance, if two messages differ in the first bit, then they will differ in S_1 when they are decoded, i.e., these two messages cannot generate the same output. Therefore, we refer to the bits corresponding to these symbols by the term "fixed bits". The "free bits", i.e., the bits which can differ in the input, are the $2 \log k - \delta = 4$ bits corresponding to the erasure $b_{17}, b_{18}, b_{19}, b_{20}$, in addition to the $\delta = 6$ deleted bits $b_3, b_4, b_5, b_6, b_7, b_8$. The total number of "free bits" is $2 \log k = 10$, so an immediate upper bound on γ is $\gamma \leq 2^{10}$. However, these "free bits" are actually constrained by the linear equations which generate the first two parities. By analyzing these constraints, one can obtain a tighter bound on γ.

The constraints on the "free bits" are given by the following system of two linear equations in $GF(32)$,

$$\begin{cases} \alpha^2 b_3 + \alpha b_4 + b_5 + \alpha^4 b_6 + \alpha^3 b_7 + \alpha^2 b_8 \\ \qquad + \alpha^3 b_{17} + \alpha^2 b_{18} + \alpha b_{19} + b_{20} = p_1', \\ \\ \alpha^2 b_3 + \alpha b_4 + b_5 + \alpha(\alpha^4 b_6 + \alpha^3 b_7 + \alpha^2 b_8) \\ \qquad + \alpha^3(\alpha^3 b_{17} + \alpha^2 b_{18} + \alpha b_{19} + b_{20}) = p_2', \end{cases} \tag{15}$$

where $p_1', p_2' \in GF(32)$ are obtained by the difference between the first and the second parity (respectively) and the part corresponding to the "free bits". To upper bound γ, we upper bound the number of solutions of the system given by (15)*. Equation* (15) *can be written as follows*

$$\begin{cases} B_1 + B_2 + B_3 = p_1', \\ B_1 + \alpha B_2 + \alpha^3 B_3 = p_2', \end{cases} \tag{16}$$

where B_1, B_2 and B_3 are three symbols in $GF(32)$ given by

$$B_1 = \alpha^2 b_3 + \alpha b_4 + b_5, \tag{17}$$

$$B_2 = \alpha^4 b_6 + \alpha^3 b_7 + \alpha^2 b_8, \tag{18}$$

$$B_3 = \alpha^3 b_{17} + \alpha^2 b_{18} + \alpha b_{19} + b_{20}. \tag{19}$$

Notice that the coefficients of B_1, B_2 and B_3 in (16) *originate from the MDS encoding vectors. Hence, if we assume that B_1 is given, then the MDS property implies that* (16) *has a unique solution for B_2 and B_3. Furthermore, since B_2 and B_3 have unique polynomial representations in $GF(32)$ of degree at most 4, then for given values of B_2 and B_3,* (18) *and* (19) *have at most one solution for $b_6, b_7, b_8, b_{17}, b_{18}, b_{19}$ and b_{20}. Therefore, an upper bound on γ is given by the number of possible choices of B_1, i.e., $\gamma \leq 2^3 = 8$.*

The analysis in Example 4 can be generalized for messages of any length k. Assume without loss of generality that $\delta = w = (m+1)\log k + 1$ deletions occur. Then, in general, the analysis yields $(m+2)\log k$ "free" bits and $k - (m+2)\log k$ "fixed" bits. Similar to (16), the $(m+2)\log k$ "free" bits are constrained by a system of $m+2$ linear equations in $GF(q)$. Note that this system of $m+2$ linear equations in $GF(q)$ does not necessarily have exactly $m+2$ variables. For instance, in Example 4, we had $m+3 = 3$ variables (B_1, B_2, B_3) in the $m+2 = 2$ equations. This happens because the $(m+2)\log k$ "free" bits may span up to $m+3$ blocks (symbols), leading to an additional symbol that is multiplied by a different MDS encoding coefficient. Therefore, since the difference between the number of symbols and number of equations is at most one, then the MDS property implies that the number of solutions of the system of equations is at most $2^{\log k} = k$, i.e., $\gamma \leq k$. Hence, from (14) we get

$$Pr\left(\boldsymbol{\mathcal{Y}_i} = \mathbf{Y} | w, p_1, ..., p_{m+2}\right) \leq \frac{k}{q^{\frac{k}{\log k} - m - 2}}, \tag{20}$$

The bound in (20) holds for an arbitrary window W. Therefore, the upper bound on the probability of decoding failure

[7]The last symbol is padded to a length of 5 bits by adding zeros.

[8]The extension field used is $GF(32)$ and has a primitive element α, with $\alpha^5 = \alpha^2 + 1$.

in (13) holds for any window location that is picked independently of the codeword. Moreover, for any given distribution on the window location (like the uniform distribution for example), we can apply the total law of probability and use the result from (20) to get

$$Pr\left(\boldsymbol{\mathcal{Y}_i} = \mathbf{Y} | p_1, ..., p_{m+2}\right) \leq \frac{k}{q^{\frac{k}{\log k} - m - 2}}, \qquad (21)$$

References

[1] S. Kas Hanna and S. El Rouayheb, "Guess & check codes for deletions, insertions, and synchronization," *arXiv preprint arXiv:1705.09569*, 2017.

[2] S. Kas Hanna and S. El Rouayheb, "Guess check codes for deletions and synchronization," in *2017 IEEE International Symposium on Information Theory (ISIT)*, pp. 2693–2697, June 2017.

[3] N. Ma, K. Ramchandran, and D. Tse, "Efficient file synchronization: A distributed source coding approach," in *Information Theory Proceedings (ISIT), 2011 IEEE International Symposium on*, pp. 583–587, IEEE, 2011.

[4] S. El Rouayheb, S. Goparaju, H. M. Kiah, and O. Milenkovic, "Synchronization and deduplication in coded distributed storage networks," *IEEE/ACM Transactions on Networking*, vol. 24, pp. 3056–3069, October 2016.

[5] S. S. T. Yazdi and L. Dolecek, "A deterministic polynomial-time protocol for synchronizing from deletions," *IEEE Transactions on Information Theory*, vol. 60, no. 1, pp. 397–409, 2014.

[6] R. Venkataramanan, V. N. Swamy, and K. Ramchandran, "Low-complexity interactive algorithms for synchronization from deletions, insertions, and substitutions," *IEEE Transactions on Information Theory*, vol. 61, no. 10, pp. 5670–5689, 2015.

[7] R. Varshamov and G. Tenengol'ts, "Correction code for single asymmetric errors," *Automat. Telemekh*, vol. 26, no. 2, pp. 286–290, 1965.

[8] V. I. Levenshtein, "Binary codes capable of correcting deletions, insertions and reversals," in *Soviet physics doklady*, vol. 10, p. 707, 1966.

[9] R. G. Gallager, "Sequential decoding for binary channels with noise and synchronization errors," tech. rep., DTIC Document, 1961.

[10] D. Cullina and N. Kiyavash, "An improvement to levenshtein's upper bound on the cardinality of deletion correcting codes," *IEEE Transactions on Information Theory*, vol. 60, pp. 3862–3870, July 2014.

[11] C. Schoeny, A. Wachter-Zeh, R. Gabrys, and E. Yaakobi, "Codes correcting a burst of deletions or insertions," *IEEE Transactions on Information Theory*, vol. 63, pp. 1971–1985, April 2017.

[12] A. S. Helberg and H. C. Ferreira, "On multiple insertion/deletion correcting codes," *IEEE Transactions on Information Theory*, vol. 48, no. 1, pp. 305–308, 2002.

[13] J. Brakensiek, V. Guruswami, and S. Zbarsky, "Efficient low-redundancy codes for correcting multiple deletions," in *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms*, pp. 1884–1892, SIAM, 2016.

[14] E. K. Thomas, V. Y. F. Tan, A. Vardy, and M. Motani, "Polar coding for the binary erasure channel with deletions," *IEEE Communications Letters*, vol. 21, pp. 710–713, April 2017.

[15] V. Levenshtein, "Asymptotically optimum binary code with correction for losses of one or two adjacent bits," *Problemy Kibernetiki*, vol. 19, pp. 293–298, 1967.

[16] L. Cheng, T. G. Swart, H. C. Ferreira, and K. A. S. Abdel-Ghaffar, "Codes for correcting three or more adjacent deletions or insertions," in *2014 IEEE International Symposium on Information Theory*, pp. 1246–1250, June 2014.

[17] P. P. Bours, "Codes for correcting insertion and deletion errors," 1994.