Codes for Channels With Segmented Edits

Mahed Abroshan
University of Cambridge
ma675@cam.ac.uk

Ramji Venkataramanan
University of Cambridge
ramji.v@eng.cam.ac.uk

Albert Guillén i Fàbregas
ICREA & Universitat Pompeu Fabra
University of Cambridge
guillen@ieee.org

***Abstract*—We consider insertion and deletion channels with the additional assumption that the channel input sequence is implicitly divided into segments such that at most one edit can occur within a segment. We further assume that there are no segment markers in the received sequence. We propose code constructions for the segmented deletion, segmented insertion, and segmented insertion-deletion channels based on subsets of VT codes chosen with pre-determined prefixes and/or suffixes. The proposed codes are zero-error, can be decoded segment-by-segment, and their rate scaling as the segment length increases is the same as that of the maximal code.**

I. Introduction

We consider the problem of constructing codes for *segmented* edit channels, where the channel input sequence is implicitly divided into disjoint segments. Each segment can undergo at most one edit, which can be either an insertion or a deletion. There are no segment markers in the received sequence.

This model, introduced by Liu and Mitzenmacher [1], is a simplified version of the general edit channel, where the insertions and deletions can be arbitrarily located in the input sequence. Constructing codes for general edit channels is well known to be challenging problem [2]–[8]. The assumption of segmented edits not only simplifies the coding problem, but is also likely to hold in many edit channels that arise in practice, e.g. in data storage.

Let us consider three examples to illustrate the model. For simplicity, we assume that the segment length, denoted by b, is 3 in each case.

1) *Segmented Deletion Channel*: Each segment can undergo at most one deletion; no insertions occur. Consider the following pair of input and output sequences:

$$X = 01\underline{1}\,10\underline{0}\,010 \;\longrightarrow\; Y = 0110010, \tag{1}$$

with the underlined bits in X being deleted by the channel to produce the output sequence Y. It is easily verified that many other input sequences could have produced the same output sequence, e.g., $01\underline{0}\,100\,\underline{0}10$, $01\underline{0}\,10\underline{1}\,010$, $011\,00\underline{0}\,1\underline{0}0$ etc. The receiver has no way of distinguishing between these candidate input sequences. In particular, despite knowing the segment length and that deletions occurred, it does not know in which two segments the deletions occurred.

This work has been funded in part by the European Research Council under ERC grant agreement 259663 and by the Spanish Ministry of Economy and Competitiveness under grant TEC2016-78434-C3-1-R.

2) *Segmented Insertion Channel*: Each segment can undergo at most one insertion; no deletions occur. The inserted bit can be placed anywhere within the segment, including before the first bit or after the last bit of the segment. For example, consider

$$X = 011\,100\,010 \;\longrightarrow\; Y = 011\underline{1}0100011\underline{0}, \tag{2}$$

with the underlined bits in Y indicating the insertions. Two inserted bits can appear between two segments whenever there is an insertion after the last bit of first segment and before the first bit of the next segment.

3) *Segmented Insertion-Deletion Channel*: This is the most general case, where a segment could undergo either an insertion or a deletion, or remain unaffected. For example, consider

$$X = 01\underline{1}\,100\,010 \;\longrightarrow\; Y = 01\underline{0}1000\underline{1}0, \tag{3}$$

with the underlined bits on the left indicating deletions, and the underlined bits on the right indicating insertions. Unlike the previous two cases, the receiver cannot even infer the exact number of edits that have occurred. In the example above, an input sequence 9 bits (three segments) long could result in a 10-bit output sequence in two different ways: either via one segment with an insertion, or via two segments with insertions and the other with a deletion.

The above examples demonstrate that one cannot reduce the problem to one of correcting one edit in a b-bit input sequence. To see this, consider the example in (1), and suppose that we used a single-deletion correcting code for each segment. Such a code would declare the first three bits of Y to be the first segment of X, which would result in incorrect decoding of the following segments.

In this paper, we construct zero-error codes for each of the three segmented edit models above. Our codes can be easily constructed even for relatively large segment sizes (several tens), and can be decoded segment-by-segment in linear time. Moreover, the proposed codes have rate of at least

$$1 - \frac{1}{b}\log_2(b+1) - \frac{\kappa}{b}, \tag{4}$$

where the constant κ equals 2 for the segmented deletion channel, 2.5 for the segmented insertion channel, and 7 for the segmented insertion-deletion channel. Thus the rate scaling for the proposed codes is the same as that of the maximal code [9, Lemma 2], with the rate penalty being at most κ/b.

The starting point for our code constructions is the family of Varshamov-Tenengolts (VT) codes [2], [10]. Each code in this family is a single-edit correcting code. In our constructions, the codewords in each segment are drawn from subsets of VT codes satisfying certain prefix/suffix conditions, which are carefully chosen to enable fast segment-by-segment VT decoding.

A. Comparison with previous work

We highlight some similarities and differences from the codes proposed by Liu and Mitzenmacher in [1] for the segmented deletion and segmented insertion channels.

Code construction: The code in [1] is a segment-by-segment code specified via sufficient conditions [1, Theorems 2.1, 2.2] that ensure that as decoding proceeds, there are at most two choices for the starting position of the next undecoded segment. Finding the maximal code that satisfies these conditions corresponds to an independent set problem, which is challenging for large b. The maximal code satisfying these conditions was reported in [1] for $b = 8, 9$. For larger b, a greedy algorithm was used to find a set of codewords satisfying the conditions. It was also suggested that one could restrict the code to a subset of VT codes that satisfy the sufficient conditions.

In comparison, our codes are directly defined as subsets of VT codes that satisfy certain simple prefix/suffix conditions; these conditions are different from those in [1]. Our conditions ensure that upon decoding each segment, there is no ambiguity in the starting position of the next segment. These subsets of VT codes are relatively simple to enumerate, so it is possible to find the largest code satisfying our conditions for b of the order of several tens. Table I lists the number of codewords per segment for the three segmented edit channels for up to $b = 24$. Another difference from the code in [1] is that for the segmented deletion and segmented insertion-deletion channels, our codebook for each segment is chosen based on the final bit of the previous segment.

Rate: The VT subsets and sufficient conditions we define allow us to obtain a lower bound of the form (4) on the rate of our code for any segment length b. Combined with the simple upper bound obtained via the rate of the maximal single-edit correcting code [9, Lemma 2], we conclude that the maximal code for the segmented edit channel has rate

$$1 - \frac{1}{b}\log_2(b+1) - O\left(\frac{1}{b}\right). \tag{5}$$

In particular, this affirmatively answers the conjecture in [1] that the rates of the maximal codes for the segmented edit channel increases with b. Our results also confirm the conjecture that the maximal codes satisfying the sufficient conditions in [1] have rate scaling optimally with b.[1]

Though the *maximal* codes satisfying the Liu-Mitzenmacher conditions have rate very close to the largest possible with segment-by-segment decoding, finding the maximal code satisfying these conditions is computationally hard, so one has to resort to greedy algorithms to construct codes for larger b. This is reflected in the rate comparison: for $b = 8, 9$, the optimal Liu-Mitzenmacher code for segmented deletions is larger than our code (12,20 vs. 8,13 codewords). However for $b = 16$, the code obtained in [1] using a greedy algorithm has 652 codewords, whereas our code has 964 codewords, as shown in Table I. For large b, our codes are nearly optimal since the rate penalty decays as κ/b.

Decoding Complexity: As segment-by-segment decoding is enforced by design, the decoding complexity grows linearly with the number of segments for both our codes and those in [1]. Within each segment, the decoding complexity of our code is linear in b. In general, for each segment, the maximal Liu-Mitzenmacher codes have to be decoded via look-up tables, in which case the complexity is exponential in b. Using subsets of VT codes was suggested in [1] as a way to reduce the decoding complexity.

Finally, we remark that codes proposed in this paper are the first for the segmented insertion-deletion model.

TABLE I: Number of codewords per segment of the proposed codes. Lower bounds computed from (11), (16), and (22) are given in brackets.

b	Deletion	Insertion	Insertion-Deletion
8	8 (8)	6 (6)	1 (1)
9	13 (13)	10 (10)	2 (1)
10	24 (24)	18 (18)	2 (1)
11	44 (43)	33 (32)	2 (2)
12	79 (79)	60 (59)	4 (3)
13	147 (147)	111 (110)	6 (5)
14	276 (274)	208 (205)	12 (9)
15	512 (512)	384 (384)	16 (16)
16	964 (964)	724 (723)	34 (31)
17	1,824 (1,821)	1,368 (1,366)	59 (57)
18	3,450 (3,450)	2,588 (2,587)	114 (108)
19	6,554 (6,554)	4,916 (4,916)	206 (205)
20	12,490 (12,484)	9,369 (9,363)	399 (391)
21	23,832 (23,832)	17,847 (17,874)	746 (745)
22	45,591 (45,591)	34,194 (34,193)	1,435 (1,425)
23	87,392 (87,382)	65,544 (65,536)	2,736 (2,731)
24	167,773 (167,773)	125,831 (125,830)	5,257 (5,243)

II. Channel Model and Preliminaries

The channel input sequence, denoted by $X = x_1x_2\cdots x_n$, is divided into k segments of b bits each. We denote the subsequence of X, from index i to index j, with $i < j$ by $X(i:j) = x_ix_{i+1}\cdots x_j$. The i-th segment of X is denoted by $S_i = s_{i,1}\cdots s_{i,b} = X\big(b(i-1)+1 : bi\big)$ for $i = 1, \ldots, k$.

In the segmented deletion channel, the channel output $Y = Y(1{:}m) = y_1\cdots y_m$, with $m \le n$ is obtained by deleting at most one bit in each segment, i.e., at most one bit in S_i, $i = 1, \ldots, k$, is deleted. Similarly, in the segmented insertion channel, the channel output $Y = y_1 \ldots y_m$, with $m \ge n$ is obtained by inserting at most one bit per segment. In the segmented insertion-deletion channel, the channel output is such that each segment S_i, $i = 1, \ldots, k$ undergoes at most one edit. In all cases, we assume that the decoder knows k and b, but not the segment boundaries. We consider coded communication

[1]This can be shown by verifying that our segmented insertion code satisfies the Liu-Mitzenmacher sufficient conditions, and noting that its rate scales optimally with b.

using a code $\mathcal{C} = \{X^{(1)}, \ldots, X^{(M)}\} \subseteq \{0,1\}^n$ of length n, M codewords and rate $R = \frac{1}{n}\log M$. The decoder produces an estimate $\hat{X}$ of the transmitted sequence. We denote the corresponding segment estimates by $\hat{S}_i = \hat{s}_{i,1} \cdots \hat{s}_{i,b}$, for $i = 1, \ldots, k$. Thus $\hat{X} = (\hat{S}_1, \ldots, \hat{S}_k)$. We consider zero-error codes that always ensure the recoverability of the transmitted sequence, i.e., codes for which $\hat{X} = X$. For simplicity, we consider binary codes, but our constructions naturally extend to larger alphabets. Levenshtein [9] showed that for any fixed number of edits e arbitrarily placed in the input sequence, the rate of the maximal code of length n that corrects e edits is $1 - \kappa_1 \frac{e \log_2 n}{n} + \kappa_2 \frac{\log_2 e!}{n}$ for sufficiently large n. Here κ_1, κ_2 are constants in the interval $[1, 2]$.

A. VT codes

In the case where $k = 1$, and thus $n = b$, there is at most one edit in the entire sequence. For this model, one can use VT codes which are zero-error single-edit correction codes [2], [10], i.e., when the transmitted codeword suffers a single insertion or a deletion, the decoder always corrects the edit. Moreover, the complexity of the VT decoding algorithm is linear in the code length b; the details of the algorithm can be found in [2].

The VT syndrome of a binary sequence $S = s_1 \ldots s_b$ is defined as

$$\mathsf{syn}(S) = \sum_{j=1}^{b} j\, s_j \pmod{b+1}. \tag{6}$$

For positive integers b and $0 \le a \le b$, we define the VT code of length b and syndrome a, denoted by

$$\mathcal{VT}_a(b) = \left\{S \in \{0,1\}^b : \mathsf{syn}(S) = a\right\} \tag{7}$$

i.e., the set of sequences S of length b that satisfy $\mathsf{syn}(S) = a$. For example,

$$\mathcal{VT}_1(3) = \left\{s_1 s_2 s_3 : \sum_{j=1}^{3} j\, s_j = 1 \bmod 4\right\} = \{100, 011\}. \tag{8}$$

The $b+1$ sets $\mathcal{VT}_a(b) \subset \{0,1\}^b$, for $0 \le a \le b$, partition the set of all sequences of length b. Each of these sets $\mathcal{VT}_a(b)$ is a single-edit correcting code. Moreover, for $0 \le a \le b$, the cardinalities of these sets satisfy [2, Corollary 2.3]

$$|\mathcal{VT}_0(b)| \ge |\mathcal{VT}_a(b)| \ge |\mathcal{VT}_1(b)|.$$

The largest of the sets $\mathcal{VT}_a(b)$, $0 \le a \le b$, will have at least $\frac{2^b}{b+1}$ sequences out of the 2^b possible. This induces a rate $R \ge 1 - \frac{1}{b}\log_2(b+1)$ for the largest of these codes. The code $\mathcal{VT}_0(b)$ has been shown to be maximal for single-edit correction for $b \le 8$, and have been conjectured to be maximal for arbitrary b [2].

III. Segmented Deletion Codes

If the decoder knew the segment boundaries, then simply using a VT code for each segment would suffice. Since the segment boundaries are not known, recall from the example in (1) that this approach is inadequate if segment-by-segment decoding is to be used. Our construction chooses a subset of a VT code for each segment, with prefixes determined by the last bit of the previous segment.

A. Code Construction

For $0 \le a \le b$, define the following sets.

$$\begin{aligned} \mathcal{A}_a^0 &\triangleq \left\{S \in \{0,1\}^b : \mathsf{syn}(S) = a,\ s_1 s_2 = 00\right\}, \\ \mathcal{A}_a^1 &\triangleq \left\{S \in \{0,1\}^b : \mathsf{syn}(S) = a,\ s_1 s_2 = 11\right\}. \end{aligned} \tag{9}$$

For $c \in \{0,1\}$, the set $\mathcal{A}_a^c \subseteq \mathcal{VT}_a(b)$ is the set of VT codewords that start with prefix cc. We now choose the sets with the largest number of codewords, i.e., we choose $\mathcal{A}_{a_0}^0$ and $\mathcal{A}_{a_1}^1$ where we define

$$a_0 = \arg\max_{0 \le a \le b} |\mathcal{A}_a^0|, \quad a_1 = \arg\max_{0 \le a \le b} |\mathcal{A}_a^1|. \tag{10}$$

By defining $M_s = \min\{|\mathcal{A}_{a_0}^0|, |\mathcal{A}_{a_1}^1|\}$, we can now construct $\mathcal{A}^0 \subseteq \mathcal{A}_{a_0}^0$ by choosing any M_s sequences from $\mathcal{A}_{a_0}^0$; similarly construct $\mathcal{A}^1 \subseteq \mathcal{A}_{a_1}^1$ by choosing any M_s sequences from $\mathcal{A}_{a_1}^1$. Note that $\mathcal{A}^0$ and $\mathcal{A}^1$ are subsets of the VT codes $\mathcal{VT}_{a_0}(b)$ and $\mathcal{VT}_{a_1}(b)$, containing sequences starting with 00 and 11, respectively.

Finally, the overall code of length $n = kb$ is constructed by choosing a codeword for each segment from either $\mathcal{A}^0$ or $\mathcal{A}^1$. The codeword for the first segment is chosen from $\mathcal{A}^0$. The codeword for segment $i = 2, \ldots, k$ is chosen as follows: if the last code bit in segment $(i-1)$ equals 0, then the codeword for segment i is chosen from $\mathcal{A}^1$; otherwise it is chosen from $\mathcal{A}^0$.

B. Rate

The overall code of length $n = kb$ has M_s^k codewords, and rate $R = \frac{1}{n}\log M_s^k = \frac{1}{b}\log M_s$. The rate can be bounded from below as

$$R \ge 1 - \frac{1}{b}\log_2(b+1) - \frac{2}{b}. \tag{11}$$

Indeed, there are 2^{b-2} binary sequences of length b whose first two bits equal 0. Each of these sequences belongs to exactly one of the sets $\mathcal{A}_0^0, \ldots, \mathcal{A}_b^0$. Therefore, the largest among these $(b+1)$ sets will contain at least $2^{b-2}/(b+1)$ sequences and thus, $|\mathcal{A}_{a_0}^0| \ge 2^{b-2}/(b+1)$. A similar argument gives the same lower bound for $|\mathcal{A}_{a_1}^1|$, hence $M_s \ge 2^{b-2}/(b+1)$. Taking logarithms gives (11).

From (11), we see that the rate penalty with respect to VT codes is at most $\frac{2}{b}$ due to the prefix of length 2. As an example, for $b = 16$ our code has 964 codewords, while the greedy algorithm described in [1], gives 740; this is reduced to 652 when the search is restricted to VT codes. More examples are reported in Table I.

C. Decoding

Thanks to the segment-by-segment code construction, decoding will also proceed segment by segment. Decoding proceeds in the following simple steps.

In order to decode segment i, for $i = 1, \ldots, k$, assume that the first $i-1$ segments have been decoded correctly. Thus the

decoder knows the correct starting position of segment i in Y; we denote it by $p_i + 1$.

By examining the last bit of segment $(i-1)$, the decoder learns the correct syndrome for the codeword in segment i, i.e., either a_0 or a_1; recall that segment 1 was drawn from $\mathcal{A}^0$. Without loss of generality, assume it is a_0; the decoding for a_1 is identical.

1) The decoder computes the VT syndrome
$$\hat{a} = \mathsf{syn}\big(Y(p_i + 1 : p_i + b)\big) \tag{12}$$
and compares it to the correct syndrome (assumed to be a_0). There are two possibilities:
   a) $\hat{a} = a_0$: The decoder concludes that there is no deletion in segment i and outputs $\hat{S}_i = Y\big(p_i + 1 : p_i + b\big)$. The starting position of the next segment in Y is $p_i + b + 1$.
   b) $\hat{a} \neq a_0$: The decoder knows there is a deletion in the first segment and feeds $Y\big(p_i + 1 : p_i + b - 1\big)$ to the VT decoder to recover the codeword. The output of the VT decoder is the decoded segment $\hat{S}_i$. The starting position of the next segment in Y is $p_i + b$.
2) The decoder now checks the last bit of the decoded segment $\hat{s}_{i,b}$. If $\hat{s}_{i,b} = 0$, the decoder knows that segment $(i+1)$ has been drawn from $\mathcal{A}^1$; otherwise it has been drawn from $\mathcal{A}^0$. Thus the decoder is now ready to decode segment $(i+1)$.

IV. Segmented Insertion Codes

A. Code Construction

As in the deletion case, we define a subset of VT codewords such that upon decoding a segment, there is no ambiguity in the starting position of the next segment. We define the following set of sequences
$$\begin{aligned}\mathcal{A}_a \triangleq \big\{S \in \{0,1\}^b : \mathsf{syn}(S) = a, S \neq 011\cdots1,\\ s_1 s_2 = 01, s_3 s_4 \neq 01\big\}\end{aligned} \tag{13}$$
and
$$a_0 = \arg\max_{0 \le a \le b} |\mathcal{A}_a|. \tag{14}$$
Similarly to the previous section, the sets $\mathcal{A}_a \subseteq \mathcal{VT}_a(b)$ are sets of VT codewords with a prefix of a certain form. Our code is thus the maximal code in this family, i.e., $\mathcal{C} = \mathcal{A}^k_{a_0}$. In contrast to the deletion case, the codeword for each segment is drawn from the same set $\mathcal{A}_{a_0}$.

In order to find the size of the code, we use similar arguments to those in the previous section. There are 2^{b-2} sequences with prefix 01, out of which 2^{b-4} are removed because they have prefix 0101; $01\cdots1$ is excluded from $\mathcal{A}_a$ by construction. Each of the $2^{b-2} - 2^{b-4} - 1$ sequences belong to exactly one of the sets $\mathcal{A}_0, \ldots, \mathcal{A}_b$. Therefore, the largest of these $b+1$ sets will have size at least
$$|\mathcal{A}_{a_0}| \ge \frac{2^{b-2} - 2^{b-4} - 1}{b+1}. \tag{15}$$
This yields the following lower bound for the rate for $b \ge 6$:
$$R \ge 1 - \frac{1}{b}\log_2(b+1) - \frac{2.5}{b}. \tag{16}$$
Hence the rate penalty is at most $\frac{2.5}{b}$ due to the added constraints on the prefix.

B. Decoding

Decoding proceeds on a segment-by-segment basis, and as in the case of deletions, the code structure ensures that before decoding segment i, the previous $(i-1)$ segments have been correctly decoded. Thus the decoder knows the correct starting position of segment i in Y; as before, denote it by $p_i + 1$.

1) The decoder computes the VT syndrome
$$\hat{a} = \mathsf{syn}\big(Y(p_i + 1 : p_i + b)\big) \tag{17}$$
and compares it to the correct syndrome a_0. There are two possibilities:
   a) $\hat{a} \neq a_0$: The decoder knows that there has been an insertion in this segment and feeds $Y\big(p_i+1 : p_i+b+1\big)$ to the VT decoder to recover the codeword. The output of the VT decoder is the decoded segment $\hat{S}_i$. The decoder proceeds decoding segment $i+1$, skipping step 2. The starting position in Y for decoding segment $i+1$ is $p_i + b + 2$.
   b) $\hat{a} = a_0$: The decoder concludes that there is no insertion in $Y\big(p_i + 1 : p_i + b\big)$ and outputs $\hat{S}_i = Y\big(p_i + 1 : p_i + b\big)$.
2) If case 1.b) holds, the decoder has to check whether y_{p_i+b+1} could be an inserted bit at the very end of the segment. To this end, the $Y(p_i + b + 1 : p_i + b + 4)$ is checked against the prefix conditions for segment $i+1$ set in $\mathcal{A}_{a_0}$.
   a) If $y_{p_i+b+1}y_{p_i+b+2} \neq 01$: the decoder understands that there is an irregularity caused by either an insertion in y_{p_i+b+1}, or in y_{p_i+b+2} or both. Therefore it deletes y_{p_i+b+1} and proceeds to decode segment $i+1$ starting from y_{p_i+b+2}.
   b) If $y_{p_i+b+1}y_{p_i+b+2} = 01$, $y_{p_i+b+3}y_{p_i+b+4} \neq 01$, then y_{p_i+b+1} is the correct start of segment $i+1$.
   c) If $y_{p_i+b+1}y_{p_i+b+2} = 01$, $y_{p_i+b+3}y_{p_i+b+4} = 01$: In this case, the decoder needs to decide among three alternatives by decoding segment $i+1$:
      i) $y_{p_i+b+3} = 0$ is an inserted bit in segment $i+1$ and no inserted bit in segment i; let $\tilde{Y}_1 = y_{p_i+b+1}y_{p_i+b+2}y_{p_i+b+4}\cdots y_{p_i+2b+1}$ denote the length b sequence resulting from deleting y_{p_i+b+3} from the received sequence. If $\mathsf{syn}(\tilde{Y}_1) = a_0$ then $\hat{S}_{i+1} = \tilde{Y}_1$.
      ii) $y_{p_i+b+4} = 1$ is an inserted bit in segment $i+1$ and no inserted bit in segment i; let $\tilde{Y}_2 = y_{p_i+b+1}y_{p_i+b+2}y_{p_i+b+3}y_{p_i+b+5}\cdots y_{p_i+2b+1}$ denote the length b sequence resulting from deleting y_{p_i+b+4} from the received sequence. If $\mathsf{syn}(\tilde{Y}_2) = a_0$ then $\hat{S}_{i+1} = \tilde{Y}_2$.
      iii) $y_{p_i+b+1} = 0$, $y_{p_i+b+2} = 1$ are inserted bits in segments i and $i+1$, respectively; let $\tilde{Y}_3 = y_{p_i+b+3}y_{p_i+b+4}\cdots y_{p_i+2b+2}$ denote the length b sequence resulting from deleting y_{p_i+b+1}, y_{p_i+b+2}

from the received sequence. If $\mathsf{syn}(\tilde{Y}_3) = a_0$ then $\hat{S}_{i+1} = \tilde{Y}_3$.

When $Y(bi+1:bi+4) = 0101$, we now show that the three cases listed in step 2.c) are mutually exclusive, and hence only one of them will give a matching VT syndrome. What needs to be checked is that the syndromes of $\tilde{Y}_1, \tilde{Y}_2, \tilde{Y}_3$ will all be different. From the very properties of VT codes we know that $\mathsf{syn}(\tilde{Y}_1) \neq \mathsf{syn}(\tilde{Y}_2)$. Now find that

$$\begin{aligned}
&\mathsf{syn}(\tilde{Y}_1) - \mathsf{syn}(\tilde{Y}_3) \quad (\mathrm{mod}(b+1)) \\
&= \sum_{j=1}^{b} j\,\tilde{y}_{1,j} - \sum_{j=1}^{b} j\,\tilde{y}_{3,j} \quad (\mathrm{mod}(b+1)) \\
&= 5 + \sum_{j=p_i+b+5}^{p_i+2b+1} y_j - 2 - b y_{p_i+2b+2} \quad (\mathrm{mod}(b+1)) \\
&= 3 + w_H\big(Y(p_i+b+5:p_i+2b+1)\big) + y_{p_i+2b+2} \\
&\qquad\qquad (\mathrm{mod}(b+1)) \\
&\neq 0
\end{aligned} \tag{18}$$

where $w_H(Z)$ denotes the Hamming weight of sequence Z. The last step of (18) holds because

$$3 + w_H\big(Y(p_i+b+5:p_i+2b+1)\big) + y_{p_i+2b+2} \ (\mathrm{mod}(b+1))$$

can equal to 0 only if $w_H\big(Y(p_i+b+5{:}p_i+2b+1)\big) = b-3$ and $y_{p_i+2b+2} = 1$, implying that both $\tilde{Y}_1 = \tilde{Y}_3 = 011\cdots1$. Since this sequence has been explicitly excluded from the codebook, we always have strict inequality, and hence $\mathsf{syn}(\tilde{Y}_1) \neq \mathsf{syn}(\tilde{Y}_3)$. Furthermore, since

$$\mathsf{syn}(\tilde{Y}_2) - \mathsf{syn}(\tilde{Y}_3) = \mathsf{syn}(\tilde{Y}_1) - \mathsf{syn}(\tilde{Y}_3) - 1 \tag{19}$$

is always non-zero, we conclude that there is no ambiguity at the decoder .

V. Segmented Insertion-Deletion Codes

A. Code Construction

Since we now have both insertion and deletions, the decoder must first identify the type of edit in a segment prior to correcting it. Define the following sets:

$$\begin{aligned}
\mathcal{A}_a^0 \triangleq \big\{ S \in \{0,1\}^b : \mathsf{syn}(S) = a,\ s_1s_2s_3s_4s_5 = 00111, \\
s_{b-2} = s_{b-1} = s_b \big\} \\
\mathcal{A}_a^1 \triangleq \big\{ S \in \{0,1\}^b : \mathsf{syn}(S) = a,\ s_1s_2s_3s_4s_5 = 11000, \\
s_{b-2} = s_{b-1} = s_b \big\}.
\end{aligned}$$

As in previous sections, these are subsets of VT codewords with certain constraints; in this case, in order to be able to identify the edit type, both prefix and suffix constraints have been added. Based on the above sets, we further define

$$a_0 = \underset{0\le a\le b}{\arg\max} |\mathcal{A}_a^0|, \quad a_1 = \underset{0\le a\le b}{\arg\max} |\mathcal{A}_a^1| \tag{20}$$

and $M_s = \min\{a_0, a_1\}$. We construct the sets $\mathcal{A}^0, \mathcal{A}^1$ by choosing M_s sequences from $\mathcal{A}_{a_0}^0, \mathcal{A}_{a_1}^1$, respectively. Finally, the overall code of length $n = kb$ is constructed by choosing a codeword for each segment from either $\mathcal{A}^0$ or $\mathcal{A}^1$. The codeword for the first segment is chosen from $\mathcal{A}^0$; if the last bit of segment $i-1$ is 0, then the codeword for segment $i = 2, \ldots, k$ is drawn from $\mathcal{A}^1$ and otherwise from $\mathcal{A}^0$.

The size and rate are lower-bounded using the same arguments as in the previous sections. For $b \geq 7$, we obtain

$$M_s \geq \frac{2^{b-7}}{b+1} \tag{21}$$

which yields a rate lower bound given by

$$R \geq 1 - \frac{1}{b}\log(b+1) - \frac{7}{b}. \tag{22}$$

Due to the prefix and suffix constraints, our segmented insertion-deletion codes have a rate penalty of at most $\frac{7}{b}$.

B. Decoding

As in the previous two cases, decoding proceeds segment-by segment. The decoder uses the suffix and prefix conditions to infer the type of edit in the current segment and ensure the correct starting position for the next segment. There are several cases to be considered. Due to space constraints, we refer the reader to [11] for a detailed description of the decoder.

References

[1] Z. Liu and M. Mitzenmacher, "Codes for deletion and insertion channels with segmented errors," *IEEE Trans on Inf. Theory*, vol. 56, no. 1, pp. 224–232, 2010.

[2] N. J. A. Sloane, "On single-deletion-correcting codes," in *Codes and Designs, Ohio State University (Ray-Chaudhuri Festschrift)*, pp. 273–291, 2000. Online: https://arxiv.org/abs/math/0207197.

[3] M. C. Davey and D. J. C. MacKay, "Reliable communication over channels with insertions, deletions, and substitutions," *IEEE Trans. Inf. Theory*, vol. 47, no. 2, pp. 687–698, 2001.

[4] E. A. Ratzer, "Marker codes for channels with insertions and deletions," *Ann. Telecommn.*, vol. 60, no. 1, pp. 29–44, 2005.

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

[6] K. A. Abdel-Ghaffar, F. Paluncic, H. C. Ferreira, and W. A. Clarke, "On Helberg's generalization of the Levenshtein code for multiple deletion/insertion error correction," *IEEE Trans. Inf. Theory*, vol. 58, no. 3, pp. 1804–1808, 2012.

[7] A. A. Kulkarni and N. Kiyavash, "Nonasymptotic upper bounds for deletion correcting codes," *IEEE Trans. Inf. Theory*, vol. 59, no. 8, pp. 5115–5130, 2013.

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

[9] V. I. Levenshtein, "Binary codes capable of correcting deletions, insertions and reversals," *Doklady Akademii Nauk SSSR*, vol. 163, no. 4, pp. 845–848, 1965. (in Russian), English Translation in *Soviet Physics Dokl.*, (No. 8, 1966), 707-710.

[10] R. R. Varshamov and G. M. Tenengolts, "Codes which correct single asymmetric errors," *Automatica i Telemekhanica*, vol. 26, no. 2, pp. 288–292, 1965. (in Russian), English Translation in *Automation and Remote Control*, (26, No. 2, 1965), 286-290.

[11] M. Abroshan, R. Venkataramanan, and A. Guillén i Fàbregas, "Codes for channels with segmented edits," 2017. (Online) http://arxiv.org/abs/1701.06341.