Codes in the Damerau Distance for DNA Storage

Ryan Gabrys*[†], Eitan Yaakobi [‡], and Olgica Milenkovic*
*ECE Department, University of Illinois, Urbana-Champaign [†]Spawar Systems Center, Pacific [‡]Technion

***Abstract*—We introduce the new problem of code design in the Damerau metric. The Damerau metric is a generalization of the Levenshtein distance which also allows for adjacent transposition edits. We first provide constructions for codes that may correct either a single deletion or a single adjacent transposition and then proceed to extend these results to codes that can simultaneously correct a single deletion and multiple adjacent transpositions. Bounds on the size of the codes and accompanying decoding algorithms are presented as well.**

I. Introduction

The edit distance is a measure of similarity between two strings evaluated based on the minimum number of operations required to transform one string into the other. If the operations are confined to symbol deletions, insertions and substitutions, the distance of interest is the Levenshtein distance [10]. The Levenshtein distance has found numerous applications in bioinformatics, where a weighted version of this metric is used to assess the similarity of DNA strings and construct phylogenetic trees [8], and natural language processing, where the distance is used to model typing or spelling errors and provide automated word corrections [2]. In parallel to the work on developing efficient algorithms for computing the edit distance and performing alignments of large number of strings, a long line of results were reported on the topic of designing codes in the Levenshtein distance. Classical derivations of upper bounds by Levenshtein [10] and single-deletion correcting code constructions by Varshamov and Tenengol'ts [14], [15] established the framework for studying many challenging problems in optimal code design for this metric [1], [6], [13].

The Damerau distance is an extension of the Levenshtein distance which allows one to account for adjacent transposition edits as well. Despite this apparent interest in coding for edit channels, the problem of designing codes in the Damerau distance was not analyzed before. A possible reason for this lack of interest in the Damerau distance may be attributed to the fact that not many models for practical channels involve adjacent transposition errors, and even if they do so, they tend not to allow for user-selected messages. Our motivating application for studying codes in the Damerau distance is the emergent paradigm of DNA-based storage [3], [5], [17], [18]. In these systems, media degradation arises due to DNA aging caused by metabolic and hydrolitic processes, or more precisely, by exposure to standard or increased level radiation, humidity, and high temperatures. As an example, human cellular DNA undergoes anywhere between 10-50 breakages in a cell cycle [16]. These DNA breakages or symbol/block of symbol deletions result in a changed structures of the string: if a string breaks in two places, which is the most likely scenario, either the sequence reattaches itself without resulting in structural damage, reattaches itself in the opposite direction, resulting in what is call a *reversal error*, or the broken string degrades, resulting in a bursty deletion; if a string breaks in three positions, which is the second most likely breakage scenario, either the adjacent broken blocks exchange positions or one or both block disintegrate leading to a bursty deletion. It is the latter scenario that motivates the study of channels in which adjacent blocks of symbols may be exchanges or individual blocks deleted. It is straightforward to see that this editing scenario corresponds to a "block version" of the Damerau editing process. The block editing process is hard to analyze directly, and we hence propose to first study the symbol-level Damerau editing process. Extensions to the block model will be described in the journal version of the paper.

The paper is organized as follows. Section II contains the problem statement and relevant notation. Section III contains an analysis of the code design procedure for single deletion or single adjacent transposition correction. Section IV contains our main contributions: an order optimal code construction for correcting a single deletion and a single adjacent transposition, as well as low-redundancy construction for codes correcting a single deletion and multiple adjacent transpositions. Independently on their use for DNA-based storage systems, our results shed light on the problem of *mismatched* Varshamov-Tenengol'ts (VT) decoding and run length limited VT codes.

II. Notation

We start by defining the *Damerau-Levenshtein distance* and its block-version, which arose by the works of Damerau [4] and Levenshtein [10].

Definition 1. *The* ***Damerau–Levenshtein distance*** *is a string metric, which for two strings* **a** *and* **b** *of length n_a and n_b over some finite alphabet equals the minimum number of insertions, deletions, substitutions and adjacent transposition edits needed to transform one string into the other. The* ***block Damerau–Levenshtein distance*** *with block length b is a string metric, which for two strings* **a** *and* **b** *of length n_a and n_b over some finite alphabet equals the minimum number of insertions, deletions, substitutions and adjacent transposition edits of blocks of length at most b needed to transform one string into the other.*

For simplicity, we focus on edits involving deletions and adjacent transpositions only, and with slight abuse of terminology refer to the underlying sequence comparison function as the Damerau metric, denoted by $D(\mathbf{a}, \mathbf{b})$[1]. Furthermore, we restrict our attention to binary alphabets as the generalizations of our results to larger alphabets are straightforward to obtain through Tenengol'ts up-down encoding, as described in [9], [11]. Details of the approach will be provided in the full version of the paper.

For a vector $\boldsymbol{x} \in \mathbb{F}_2^n$, let $\mathcal{B}(\boldsymbol{x})$ denote the set of vectors that may be obtained from $\boldsymbol{x}$ by either at most one single deletion or at most one single adjacent transposition. Note that the size of $\mathcal{B}(\boldsymbol{x})$ is $2r(\boldsymbol{x})$, where $r(\boldsymbol{x})$ is the number of runs in $\boldsymbol{x}$, i.e., the smallest number of nonoverlapping substrings involving the same symbol that "covers" the sequence.

[1]Since we only consider deletions, the function is not a metric.

Example 1. Suppose that $\boldsymbol{x} = (0,0,1,1,0) \in \mathbb{F}_2^n$. Then,

$$\begin{aligned}\mathcal{B}(\boldsymbol{x}) = \{&(0,1,1,0),(0,0,1,0),(0,0,1,1),\\ &(0,0,1,1,0),(0,1,0,1,0),(0,0,1,0,1)\}.\end{aligned}$$

The derivative of $\boldsymbol{x}$, denoted by $\partial(\boldsymbol{x}) = \boldsymbol{x}'$ is a vector given by $\boldsymbol{x}' = (x_1, x_2+x_1, x_3+x_2, \ldots, x_n+x_{n-1})$. Observe that the mapping between $\boldsymbol{x}$ and $\boldsymbol{x}'$ is a bijection. Hence the integral $\partial^{-1}(\boldsymbol{x}) \triangleq \overline{\boldsymbol{x}}$ is well-defined for all $\boldsymbol{x} \in \mathbb{F}_2^n$. For a set $\mathcal{X} \subseteq \mathbb{F}_2^n$, we use $\mathcal{X}'$ to denote the set of derivative of vectors in $\mathcal{X}$, and similarly, we use $\overline{\mathcal{X}}$ to denote the set of integrals of vectors in $\mathcal{X}$. For two vectors $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{F}_2^n$, we let $d_H(\boldsymbol{x}, \boldsymbol{y})$ denote their Hamming distance. Furthermore, we let $\mathcal{C}_H(n,d)$ stand for any code of length n with minimum Hamming distance d, and similarly, let $\mathcal{C}_D(n)$ stand for any single-deletion-correcting code of length n.

III. Single Transposition or Deletion Correcting Codes

We start by describing a general construction for single transposition *or* deletion correcting codes. We then show how to use this construction in order to devise codes with near-optimal redundancy.

Construction 1 Let $\mathcal{C}_H(n,3)$ be a single-error-correcting code and as before, let $\mathcal{C}_D(n)$ be a single-deletion-correcting code. We define a code $\mathcal{C}_{T\vee D}(n)$ capable of correcting one transposition (T) or ($\vee$) one deletion (D) as follows:

$$\mathcal{C}_{T\vee D}(n) = \{\boldsymbol{x} \in \mathbb{F}_2^n : \boldsymbol{x} \in \mathcal{C}_D(n), \overline{\boldsymbol{x}} \in \mathcal{C}_H(n,3)\}.$$

Note that the code $\mathcal{C}_{T\vee D}(n)$ consists of codewords that belong to a single-deletion-correcting code and have integrals that belong to a single-error-correcting code. The correctness of this construction is proven next.

Lemma 2. *The code $\mathcal{C}_{T\vee D}(n)$ from Construction 1 is a single transposition or deletion correcting code.*

Proof: We prove this claim by showing that for all $\boldsymbol{x} \in \mathcal{C}_{T\vee D}(n)$, one can uniquely recover $\boldsymbol{x}$ from any $\boldsymbol{z} \in \mathcal{B}(\boldsymbol{x})$.

Assume first that $\boldsymbol{z} \in \mathbb{F}_2^{n-1}$, so that $\boldsymbol{z}$ is the result of a single deletion occurring in $\boldsymbol{x}$. Since $\boldsymbol{x} \in \mathcal{C}_D(n)$, one may apply the decoder of the code $\mathcal{C}_D(n)$ to successfully recover $\boldsymbol{x} \in \mathcal{C}_{T\vee D}(n)$.

Now assume that $\boldsymbol{z} \in \mathbb{F}_2^n$, so that $\boldsymbol{z}$ is the result of at most one single transposition occurring in $\boldsymbol{x}$. We show that $d_H(\overline{\boldsymbol{x}}, \overline{\boldsymbol{z}}) \leqslant 1$. Then since $\overline{\boldsymbol{x}}$ belongs to a code with minimum Hamming distance 3, the vector $\overline{\boldsymbol{x}}$ can be uniquely determined based on $\overline{\boldsymbol{z}}$. Note that since the mapping ∂ is injective, $d_H(\overline{\boldsymbol{x}}, \overline{\boldsymbol{z}}) = 0$ if and only if $\overline{\boldsymbol{x}} = \overline{\boldsymbol{z}}$.

Assume that the transmitted word $\boldsymbol{x}$ was subjected to one adjacent transposition involving the ith and $(i+1)$th bits, so that $x_i \neq x_{i+1}$ and $\boldsymbol{z} = (x_1, \ldots, x_{i-1}, x_{i+1}, x_i, x_{i+2}, \ldots, x_n)$. First, we compute the integral $\overline{\boldsymbol{z}}$ as

$$\overline{\boldsymbol{z}} = (z_1, z_2+z_1, z_3+z_2+z_1, \ldots, \sum_{j=1}^{n} z_j) = (\overline{z}_1, \ldots, \overline{z}_n).$$

Let $\overline{\boldsymbol{x}} = (\overline{x}_1, \ldots, \overline{x}_n)$. Then, clearly $(\overline{x}_1, \ldots, \overline{x}_{i-1}) = (\overline{z}_1, \ldots, \overline{z}_{i-1})$. Furthermore,

$$\overline{z}_i = \sum_{j=1}^{i-1} x_j + x_{i+1} = \sum_{j=1}^{i-1} x_j + (1+x_i) = \bar{x}_i + 1,$$

and for any $k \geqslant i+1$, $\overline{z}_k = \sum_{j=1}^{i-1} x_j + x_{i+1} + x_i + \sum_{j=i+2}^{n} x_j = \overline{x}_k$, so that $d_H(\overline{\boldsymbol{x}}, \overline{\boldsymbol{z}}) = 1$ as desired. ■

Note that we did not specify in Construction 1 which specific codes to use. An obvious choice would be to use a Hamming code as the single-error-correcting code and the Varshamov-Tenengol'ts (VT) code as the single-deletion code [10], or any coset of these codes. Since the cosets of these codes cover $\mathbb{F}_2^n$, one can see that there exists a code with redundancy at most $2\log(n+1)$. We show next how to improve this result by constructing one code that can serve both as a single deletion-correcting for $\boldsymbol{x}$ and a single error-correcting code for $\overline{\boldsymbol{x}}$. The redundancy of this code is at most $\log(n)+3$.

Our choice of codes is as follows. Let a be a non-negative integer $0 \leqslant a \leqslant 6n-2$. For the single-deletion code, we make use of the code

$$\begin{aligned}\mathbf{Y}_D(n,a) =\{&\boldsymbol{y} \in \mathbb{F}_2^n : \sum_{i=1}^{n-1} i\, y_i +\\ &(2n-1)\, y_n \equiv a \mod (6n-3)\}.\end{aligned}$$

For the code $\mathcal{C}_H(n,3)$, we choose

$$\begin{aligned}\mathbf{Y}_H(n,a) = \Bigg\{&\boldsymbol{y} \in \mathbb{F}_2^n : \sum_{i=1}^{n-2} (2i+1)\, y_i + (2(n-1)+1)\, y_n\\ &+ (3(n-1)+1)\, y_{n-1} \equiv a \mod (6n-3)\Bigg\}.\end{aligned}$$

Claim 1. *For any vector $\boldsymbol{x} \in \mathbb{F}_2^n$, if $\boldsymbol{x}' \in \mathbf{Y}_D(n,a)$ then $\boldsymbol{x} \in \mathbf{Y}_H(n,a)$ and thus if $\boldsymbol{x} \in \mathbf{Y}_D(n,a)$ then $\overline{\boldsymbol{x}} \in \mathbf{Y}_H(n,a)$.*

According to Claim 1 and Lemma 2, the code $\mathcal{C}_{T\vee D}(n) = \mathbf{Y}_D(n,a)$ is a single transposition or deletion correcting code. We only have to show that the codes $\mathbf{Y}_D(n,a)$ and $\mathbf{Y}_H(n,a)$ have the desired properties.

Lemma 3. *The code $\mathbf{Y}_H(n,a)$ is a single-error-correcting code.*

Lemma 4. *The code $\mathbf{Y}_D(n,a)$ can correct a single deletion.*

The following corollary summarizes the main result of this section.

Corollary 5. *There exists a single transposition or deletion correcting code whose redundancy is at most* $\log(6n-3)$ *bits.*

Proof: We let $\mathcal{C}_{T\vee D}(n) = \mathbf{Y}_D(n,a)$. Since the codes $\mathcal{C}_{T\vee D}(a,n)$ for $0 \leqslant a \leqslant 6n-2$ partition the space $\mathbb{F}_2^n$, we easily see that its redundancy is at most $\log(6n-3)$. ■

Note that every single transposition or deletion correcting code is also a single-deletion code. A lower bound on the redundancy of the latter code is $\log(n)$ [7], so that the difference between the redundancy of our deletion/adjacent transposition codes and the optimal single deletion redundancy is at most $\log 6$ bits.

IV. Codes Correcting Deletions and Adjacent Transpositions

We now turn our attention to the significantly more challenging task of constructing codes that can correct both deletions and adjacent transpositions simultaneously. Our main result is a construction of a code capable of correcting a single deletion along with multiple adjacent transpositions. At the end of this section, we present an improved construction for the special case of a single deletion and a single transposition.

We first introduce some useful notation. Let $\mathcal{B}_{(T,\ell)}(\boldsymbol{x})$ denote the set of vectors that may be obtained by applying at most ℓ adjacent transpositions (T) to $\boldsymbol{x}$. We define $\mathcal{B}_{(T,\ell)}(\boldsymbol{x}) \triangleq \underbrace{\mathcal{B}_{(T,1)}(\cdots(\mathcal{B}_{(T,1)}(}_{\ell \text{ times}}\boldsymbol{x})\cdots))$. Let $\mathcal{B}_{(T,\ell),D}(\boldsymbol{x})$ denote the set of vectors that may be obtained from $\boldsymbol{x}$ by at most ℓ adjacent transpositions followed by a single deletion. Let $\mathcal{B}_D(\boldsymbol{x})$ be the set of words that may be obtained by applying a single deletion to $\boldsymbol{x}$. With a slight abuse of notation, we use the same symbol $\mathcal{B}$ independent on the the argument of the set being a word or a collection of words. In the latter case, the set $\mathcal{B}$ equals the union of the corresponding sets of individual words in the argument. The next example illustrates the relevant notation.

Example 2. Suppose that $\boldsymbol{x} = (0,0,1,1,0)$. Then, $\mathcal{B}_{(T,1)}(\boldsymbol{x}) = \{(0,0,1,1,0),\ (0,1,0,1,0),\ (0,0,1,0,1)\}$, $\mathcal{B}_D(\boldsymbol{x}) = \{(0,1,1,0), (0,0,1,0), (0,0,1,1)\}$, $\mathcal{B}_{(T,1),D}(\boldsymbol{x}) = \{(0,1,1,0), (0,0,1,0), (0,0,1,1), (1,0,1,0), (0,1,0,1), (0,1,0,0), (0,0,0,1)\}$.

Lemma 6. *For any $\boldsymbol{x} \in \mathbb{F}_2^n$,*

$$\mathcal{B}_{(T,\ell),D}(\boldsymbol{x}) = \mathcal{B}_D(\mathcal{B}_{(T,\ell)}(\boldsymbol{x})) = \mathcal{B}_{(T,\ell)}(\mathcal{B}_D(\boldsymbol{x})).$$

As a consequence of the previous lemma, we may hence assume that the deletion always occurs after the adjacent transposition. We say that a code $\mathcal{C}$ can correct ℓ adjacent transpositions and a single deletion, and refer to it as a *ℓ-TD code* if for any two different codewords $\boldsymbol{x}, \boldsymbol{y} \in \mathcal{C}$, $\mathcal{B}_{(T,\ell),D}(\boldsymbol{x}) \cap \mathcal{B}_{(T,\ell),D}(\boldsymbol{y}) = \emptyset$. Our code construction and the ideas behind our approach are best explained by describing the decoding procedure.

Suppose that the code $\mathcal{C}_{T\wedge D}(n,\ell)$ is an ℓ-TD code, which is a subcode of a single-deletion-correcting code. Assume also that $\boldsymbol{x} \in \mathcal{C}_{T\wedge D}(n,\ell)$ was transmitted and the vector $\boldsymbol{y}$ was received, where $\boldsymbol{y}$ is the result of at most ℓ transpositions followed by a single deletion occurring to $\boldsymbol{x}$. The simplest idea to pursue is to try to correct the single deletion by naively applying the decoder for the chosen constituent single-deletion code. Clearly, such a decoder will produce an erroneous result due to the presence of the adjacent transposition errors. It is therefore important to construct the code $\mathcal{C}_{T\wedge D}(n,\ell)$ in such a way that the result of the "mismatched" deletion correction $\widehat{\boldsymbol{x}}$, obtained from $\boldsymbol{y}$, is easy to characterize and contains only a limited number of errors that may be corrected to recover $\boldsymbol{x} \in \mathcal{C}_{T\wedge D}(n,\ell)$ from $\widehat{\boldsymbol{x}}$. To this end, define the following code

$$\mathcal{C}_{VT}(n,a,\ell) = \{\boldsymbol{x} \in \mathbb{F}_2^n : \sum_{i=1}^{n} i \cdot x_i \equiv a \bmod (n+2\ell+1)\}.$$

Since the code is a VT code, the decoder $\mathcal{D}_{VT,n,\ell}$ for $\mathcal{C}_{VT}(n,a,\ell)$ can correct a single deletion occurring in a codeword $\boldsymbol{x} \in \mathcal{C}_{VT}(n,a,\ell)$ [14].

As before, and for the special case of VT codes, assume that $\widehat{\boldsymbol{x}}$ is the result of VT decoding the vector $\boldsymbol{y}$ where $\boldsymbol{y} \in \mathcal{B}_{(T,\ell),D}(\boldsymbol{x})$. Our first aim is to characterize the difference between $\widehat{\boldsymbol{x}}$ and $\boldsymbol{x}$, and for this purpose we use an intermediary word $\boldsymbol{y}^{(\ell)}$ that is generated from at most ℓ adjacent transpositions in $\boldsymbol{x}$, that is $\boldsymbol{y} \in \mathcal{B}_D(\boldsymbol{y}^{(\ell)})$. More precisely, we demonstrate that if both $\boldsymbol{x}, \boldsymbol{y}^{(\ell)} \in \mathcal{C}_{VT}(n,a,\ell)$, then $\mathcal{D}_{VT,n,\ell}(a,\boldsymbol{x})$ and $\mathcal{D}_{VT,n,\ell}(a,\boldsymbol{y}^{(\ell)})$ differ only in the transpositions which converted $\boldsymbol{x}$ to $\boldsymbol{y}^{(\ell)}$. On the other hand, if $\boldsymbol{x}, \boldsymbol{y}^{(\ell)}$ belong to two different VT codes (i.e. they have different values of the parameter a when VT-transformed), then $\boldsymbol{x}$ and $\widehat{\boldsymbol{x}}$ differ by at most 2ℓ adjacent transpositions. The following simple claim is a consequence of the fact that an adjacent transposition changes the VT syndrome by at most one.

Claim 2. *Suppose that $\boldsymbol{y}^{(\ell)} = (y_1^{(\ell)}, \ldots, y_n^{(\ell)}) \in \mathcal{B}_{(T,\ell)}(\boldsymbol{x})$ where $\boldsymbol{x} \in \mathbb{F}_2^n$. Then, one has $|\sum_{i=1}^n i \cdot x_i - \sum_{i=1}^n i \cdot y_i^{(\ell)}| \leqslant \ell$.*

As a consequence of the previous claim, if $\boldsymbol{x} \in \mathcal{C}_{VT}(n,a,\ell)$ and $\boldsymbol{y}^{(\ell)} \in \mathcal{B}_{(T,\ell)}(\boldsymbol{x})$, then $\boldsymbol{y}^{(\ell)} \in \mathcal{C}_{VT}(n,\hat{a},\ell)$ for some $\hat{a}$, where $|a-\hat{a}| \leqslant \ell$. The next lemma summarizes the previous discussion.

Lemma 7. *Suppose that $\boldsymbol{y}^{(\ell)} \in \mathcal{B}_{(T,\ell)}(\boldsymbol{x})$, where $\boldsymbol{x} \in \mathcal{C}_{VT}(n,a,\ell)$, and let $\boldsymbol{y} \in \mathcal{B}_D(\boldsymbol{y}^{(\ell)})$. Then, $\mathcal{D}_{VT,n,\ell}(\hat{a},\boldsymbol{y}) = \boldsymbol{y}^{(\ell)}$ for some $\hat{a}$ such that $|a-\hat{a}| \leqslant \ell$.*

Example 3. Suppose that $\boldsymbol{x} = (0,1,1,0,0,1,0,0,0,0,1,0) \in \mathcal{C}_{VT}(12,3,3)$ was transmitted and that the vector $\boldsymbol{y} = (0,1,1,0,\ 0,1,0,0,\ 1,0,0)$ was received after at most three adjacent transpositions and a single deletion. For $\boldsymbol{y}^{(3)} = (0,1,1,0,\ 0,1,0,0,\ 0,1,0,0)$ (where $\boldsymbol{y} \in \mathcal{B}_D(\boldsymbol{y}^{(3)})$, we have $\sum_{i=1}^{n-1} y_i \equiv 2 \bmod 19$. Thus, since $a=3$ and $\hat{a}=2$, we get that $|a-\hat{a}| \leqslant 1 \leqslant \ell = 3$ as desired.

Note that if we apply the decoder $\mathcal{D}_{VT,12,3}$ we get $\widehat{\boldsymbol{x}} = \mathcal{D}_{VT,12,3}(3,\boldsymbol{y}) = (0,1,1,0,0,0,1,0,0,1,0,0)$. Here, we have $\widehat{\boldsymbol{x}} = (0,1,1,0,0,0,1,0,0,1,0,0)$, and $\boldsymbol{x} = (0,1,1,0,0,1,0,0,0,0,1,0)$.

We want to characterize next the difference between $\mathcal{D}_{VT,n,\ell}(a,\boldsymbol{y})$ and $\mathcal{D}_{VT,n,\ell}(\hat{a},\boldsymbol{y})$ for the case that $|a-\hat{a}| \leqslant \ell$, as the value $\hat{a}$ is unknown beforehand. Our main result may be intuitively described as follows: suppose that $\boldsymbol{y} \in \mathcal{B}_D(\boldsymbol{x})$, where $\boldsymbol{x} \in \mathcal{C}_{VT}(n,a,\ell)$ and where $\boldsymbol{y}$ is obtained by deleting the kth bit, x_k, from $\boldsymbol{x}$ and where the value of x_k is known to the decoder. Assume that $\widehat{\boldsymbol{x}} = \mathcal{D}_{VT,n,\ell}(a+v,\boldsymbol{y})$, for some offset v, is obtained by inserting the bit x_k in $\boldsymbol{y}$. Then, if $x_k = 0$, we may obtain $\boldsymbol{x}$ from $\widehat{\boldsymbol{x}}$ by sliding the inserted bit to the left/right using a series of adjacent transposition operations past at most v ones. Otherwise, if $x_k = 1$, then we can obtain $\boldsymbol{x}$ from $\widehat{\boldsymbol{x}}$ by sliding the inserted bit to the left/right past at most v zeros. The next lemma rigorously summarizes this observation.

Lemma 8. *Suppose that $\boldsymbol{y}$ is the result of a single deletion occurring in $\boldsymbol{x} \in \mathcal{C}_{VT}(n,a,\ell)$ at position k. Given k, let $v_L = |\{j \in [n] : j < k, x_j = 1\}|$ and $v_R = |\{j \in [n] : j > k, x_j = 1\}|$. Then,*

1) *If $x_k = 0$, then for all $v \in \{-v_R, -v_R+1, \ldots, v_L\}$, one may obtain $\mathcal{D}_{VT,n,\ell}(a+v,\boldsymbol{y})$ by inserting the symbol 0 in $\boldsymbol{y}$ immediately after the $(v_L - v)$-th one.*
2) *If $x_k = 1$, then for all $v \in \{-(k-1)+v_L, -k+v_L+2, \ldots, (n-k)-v_R\}$, one may obtain $\mathcal{D}_{VT,n,\ell}(a+v,\boldsymbol{y})$ by inserting the symbol 1 in $\boldsymbol{y}$ immediately after the $(v+k-v_L-1)$-th zero.*

Example 4. Suppose that $\boldsymbol{x} = (0,1,1,0,0,1,0,0,0,0,1,0) \in \mathcal{C}_{VT}(12,3,3)$, and that $\hat{\boldsymbol{x}} = \mathcal{D}_{VT,n,\ell}(3,\boldsymbol{y})$, was obtained by VT decoding $\boldsymbol{y} = (0,1,1,0,1,0,0,0,0,1,0)$. For $v = 2$, one has $\mathcal{D}_{VT,n,\ell}(5,\boldsymbol{y}) = (0,0,1,1,0,1,0,0,0,0,1,0)$, whereas for $v=-1$, one has $\mathcal{D}_{VT,n,\ell}(2,\boldsymbol{y}) = (0,1,1,0,1,0,0,0,0,0,1,0)$.

Next, suppose that $\boldsymbol{y} = (0,1,1,0,0,0,0,0,0,1,0)$ – $\boldsymbol{y}$ is the result of deleting the third 1 at position $k=6$ from $\boldsymbol{x} = (0,1,1,0,0,1,0,0,0,0,1,0)$. In this case, choosing $v=3$ gives $\mathcal{D}_{VT,n,\ell}(6,\boldsymbol{y}) = (0,1,1,0,0,0,0,0,1,0,1,0)$, while $v=-2$ gives $\mathcal{D}_{VT,n,\ell}(1,\boldsymbol{y}) = (0,1,1,1,0,0,0,0,0,0,1,0)$.

Proof of Lemma 8: Suppose first that $\boldsymbol{y}$ is the result of deleting a zero from $\boldsymbol{x} \in \mathcal{C}_{VT}(n,a,\ell)$. Let $a' \equiv a - \sum_{i=1}^{n-1} i \cdot y_i \bmod (n+2\ell+1)$. The decoder $\mathcal{D}_{VT,n,\ell}$ for $\mathcal{C}_{VT}(n,a,\ell)$ produces the vector $\widehat{\boldsymbol{x}} \in \mathcal{C}_{VT}(n,a,\ell)$ by inserting a zero into the first position k' that has a' ones to the right of it. If $x_k = 0$, then clearly $a' = v_R$, $k' = k$, and the decoder correctly outputs $\boldsymbol{x}$ so that $\widehat{\boldsymbol{x}} = \boldsymbol{x}$. If the decoder $\mathcal{D}_{VT,n,\ell}$ for $\mathcal{C}_{VT}(n,a+v,\ell)$ were applied to $\boldsymbol{y}$ instead, one would have

$$a'' \equiv a+v-\sum_{i=1}^{n-1} i\cdot y_i \bmod (n+2\ell+1) \equiv a'+v \bmod (n+2\ell+1).$$

Hence, the decoder $\mathcal{D}_{VT,n,\ell}$ for $\mathcal{C}_{VT}(n,a+v,\ell)$ would insert a zero in the vector $\boldsymbol{y}$ at the first position k'' that has $a'+v$ ones to the right of it. The claim follows by observing that the position that has $a'+v$ ones after it is in the same run as the position in $\boldsymbol{y}$ with $(v_L - v)$ ones preceding it.

Suppose now that $\boldsymbol{y}$ is the result deleting a one from $\boldsymbol{x} \in \mathcal{C}_{VT}(n,a,\ell)$. Let $a' \equiv a - \sum_{i=1}^{n-1} i \cdot y_i \bmod (n+2\ell+1)$. The decoder $\mathcal{D}_{VT,n,\ell}$ for $\mathcal{C}_{VT}(n,a,\ell)$ produces the vector $\widehat{\boldsymbol{x}} \in \mathcal{C}_{VT}(n,a,\ell)$ by inserting a one into the first position k' that has $a'-k'$ ones to the right of it. If $x_k = 1$, then clearly $k' = k$ and the decoder correctly outputs $\boldsymbol{x}$, and hence $\widehat{\boldsymbol{x}} = \boldsymbol{x}$.

If the decoder $\mathcal{D}_{VT,n,\ell}$ for $\mathcal{C}_{VT}(n,a+v,\ell)$ were applied to $\boldsymbol{y}$ instead, then $a'' \equiv a'+v \bmod (n+2\ell+1)$ as before. The decoder $\mathcal{D}_{VT,n,\ell}$ for $\mathcal{C}_{VT}(n,a+v,\ell)$ would insert a one in the vector $\boldsymbol{y}$ into the first position k'' that has $a'+v-k''$ ones to the right of it. This produces a vector $\widehat{\boldsymbol{x}}$. Since the first position k'' in $\boldsymbol{y}$ with $a'+v-k''$ ones to the right is the same as the first position in $\boldsymbol{y}$ following $(v+k-v_L-1)$ zeros, the claimed result also follows for $x_k = 1$. ■

The previous lemma allows us to propose a modification of a VT code capable of correcting a deletion and multiple adjacent transpositions. Furthermore, it leads to a straightforward decoding architecture for the proposed code. Formally,

$$\begin{aligned}\mathcal{C}_{VT}(n,a,b,\ell) = \{\boldsymbol{x} \in \mathbb{F}_2^n :& \\ &\sum_{i=1}^{n} i\cdot x_i \equiv a \bmod (n+2\ell+1), \\ &\sum_{i=1}^{n} x_i \equiv b \bmod 2\}.\end{aligned} \tag{1}$$

The decoder for $\mathcal{C}_{VT}(n,a,b,\ell)$, denoted by $\mathcal{D}_{VT,n,b,\ell}$, operates as follows. Suppose that $\boldsymbol{x} \in \mathcal{C}_{VT}(n,a,b,\ell)$ is transmitted and that $\boldsymbol{y} \in \mathcal{B}_{(T,\ell),D}(\boldsymbol{x})$ is received. Suppose that n_1 denotes the number of ones in $\boldsymbol{y}$. Then, for $a \in \mathbb{Z}_{n+2\ell+1}$ and $b \in \mathbb{F}_2$, $\mathcal{D}_{VT,n,b,\ell}(a,\boldsymbol{y})$ executes the next steps:

1) Set $x \equiv \sum_{i=1}^{n-1} x_i + b \bmod 2$.
2) Compute $a' \equiv a - \sum_{i=1}^{n-1} i\, y_i \bmod (n+2\ell+1)$.
3) If $x=0$ and $a' \in \{0,1,\ldots,n_1\}$, insert a zero into the first position in $\boldsymbol{y}$ that has a' ones on its right. If $a' \in \{n_1+1, n_1+2, \ldots, n_1+\ell\}$, insert a zero in the first position in $\boldsymbol{y}$. If $a' \in \{n_1+\ell+1, n_1+\ell+2, \ldots, n_1+2\ell\}$, insert a zero in the last position of $\boldsymbol{y}$.
4) If $x=1$ and $a' \in \{n_1+1, n_1+2, \ldots, n\}$, insert a one in the first position k of $\boldsymbol{y}$ that has $a'-k$ ones to its right. Otherwise, if $a' \in \{n+1, n+2, \ldots, n+\ell\}$, insert a one in the last position of $\boldsymbol{y}$. If $a' \in \{n-\ell+1, n_1-\ell+2, \ldots, n_1\}$, insert a one in the first position of $\boldsymbol{y}$.

Note that the VT decoder discussed so far aims to correct a single deletion only, but potentially in a mismatched fashion as additional adjacent transposition errors may exist. The output of this decoder has to be fed into the input of a transposition error-correcting code, and we will describe how this is accomplished after providing an illustration of the VT decoding process.

Example 5. Suppose that $\boldsymbol{x} = (0,1,1,0,0,1,0,0,0,0,1,0) \in \mathcal{C}_{VT}(12,3,0,3)$, and that $\boldsymbol{y} = (0,1,1,0,1,0,0,0,1,0,0)$ is the received word, which is the result of a deletion and a single transposition. We first apply the decoder $\mathcal{D}_{VT,12,0,3}$ to $\boldsymbol{y}$. From the first step of the procedure, we conclude that the deleted bit has value $x=0$. In the second step of decoding, we compute $a'=3$. Since $0 \leqslant a' \leqslant 4$, we have $\widehat{\boldsymbol{x}} = (0,1,0,1,0,1,0,0,0,1,0,0)$. Note that $\widehat{\boldsymbol{x}} = (0,1,0,1,0,1,0,0,0,1,0,0)$, and $\boldsymbol{x} = (0,1,1,0,0,1,0,0,0,0,1,0)$, differ in two adjacent transpositions.

The previous example illustrates that $\boldsymbol{x}$ and $\widehat{\boldsymbol{x}}$ differ in a limited number of transpositions which depends on the original number of transposition errors. In particular, for the given example, the two vectors differed in two adjacent transpositions as $\widehat{\boldsymbol{x}}$ is the result of a single deletion and a single transposition in $\boldsymbol{y}$. The next lemma gives a more precise characterization of the "distance" between $\boldsymbol{x}$ and $\widehat{\boldsymbol{x}}$.

Lemma 9. *Suppose that $\boldsymbol{y}^{(\ell)} = \mathcal{B}_{(T,\ell)}(\boldsymbol{x})$, where $\boldsymbol{x} \in \mathcal{C}_{VT}(n,a,b,\ell)$ and where $\boldsymbol{y} \in \mathcal{B}_D(\boldsymbol{y}^{(\ell)})$. Let $\widehat{\boldsymbol{x}} = \mathcal{D}_{VT,n,b,\ell}(a,\boldsymbol{y})$. Then the following statements are true:*

1) *If $\widehat{\boldsymbol{x}}$ is the result of inserting a zero in $\boldsymbol{y}$ in a position with v_R ones to the right of the inserted bit, then $\boldsymbol{y}^{(\ell)}$ can be obtained from $\boldsymbol{y}$ by inserting a zero in $\boldsymbol{y}$ in the first position with j ones to the right of it where $j \in \{v_R-\ell, v_R-\ell+1, \ldots, v_R+\ell\}$.*
2) *If $\widehat{\boldsymbol{x}}$ is the result of inserting a one in $\boldsymbol{y}$ in position k with v_R ones to the right of the inserted bit, then $\boldsymbol{y}^{(\ell)}$ can be obtained from $\boldsymbol{y}$ by inserting a one in $\boldsymbol{y}$ in the first position j with j_1 ones to the right of it where $j+j_1 \in \{k+v_R-\ell, k+v_R-\ell+1, \ldots, k+v_R+\ell\}$.*

The following corollary summarizes one of the main results of this section.

Corollary 10. *Suppose that $\boldsymbol{y} = \mathcal{B}_{(T,\ell),D}(\boldsymbol{x})$ where $\boldsymbol{x} \in \mathcal{C}_{VT}(n,a,b,\ell)$ and let $\widehat{\boldsymbol{x}} = \mathcal{D}_{VT,n,b,\ell}(a,\boldsymbol{y})$. Then $\boldsymbol{x} \in \mathcal{B}_{(T,2\ell)}(\widehat{\boldsymbol{x}})$.*

Consequently, the mismatched VT decoder increases the number of adjacent transposition errors by at most a factor of two.

Based on the results on mismatched VT decoding and Corollary 10, we are now ready to define a family of codes capable of correcting a single deletion and multiple adjacent transposition errors. Recall that given a binary word $\mathbf{x}$, its derivative $\partial(\boldsymbol{x}) = \boldsymbol{x}'$ is defined as $\boldsymbol{x}' = (x_1, x_2 + x_1, x_3 + x_2, \ldots, x_n + x_{n-1})$ and its inverse $\partial^{-1}(\boldsymbol{x}) = \overline{\boldsymbol{x}} = (x_1, x_1 + x_2, \ldots, \sum_{i=1}^{n} x_i)$. We claim that the code $\mathcal{C}_{(T,\ell)\wedge D} \subseteq \mathbb{F}_2^n$, defined as

$$\mathcal{C}_{(T,\ell)\wedge D}(n,a,b) = \{\boldsymbol{x} \in \mathbb{F}_2^n : \overline{\boldsymbol{x}} \in \mathcal{C}_H(n, 4\ell+1), \quad \boldsymbol{x} \in \mathcal{C}_{VT}(n,a,b,\ell)\}. \quad (2)$$

is an ℓ-TD code (i.e., a code capable of correcting ℓ adjacent transpositions (T,ℓ) *and* $(\wedge)$ one deletion (D).

Theorem 11. *The code $\mathcal{C}_{(T,\ell)\wedge D}(n,a,b)$ is an ℓ-TD code.*

Proof: Suppose that $\boldsymbol{y} \in \mathcal{B}_{(T,\ell),D}(\boldsymbol{x})$. We show how to recover $\boldsymbol{x}$ from $\boldsymbol{y}$. First, we determine $\widehat{\boldsymbol{x}} = \mathcal{D}_{VT,n,b,\ell}(a, \boldsymbol{y})$. From Corollary 10, we have that $\boldsymbol{x} \in \mathcal{B}_{(T,2\ell)}(\widehat{\boldsymbol{x}})$. Since $\boldsymbol{x} \in \mathcal{B}_{(T,2\ell)}(\widehat{\boldsymbol{x}})$, we have $d_H(\partial^{-1}(\widehat{\boldsymbol{x}}), \overline{\boldsymbol{x}}) \leqslant 2\ell$. Because the minimum distance of the code $\overline{\mathcal{C}}_{(T,\ell)\wedge D}(n,a,b)$ is $4\ell+1$, we can uniquely recover $\boldsymbol{x}$ from $\partial^{-1}(\widehat{\boldsymbol{x}})$. ■

Corollary 12. *There exists an ℓ-TD code which redundancy at most $(2\ell+1)\log(n)$ bits.*

Lastly in this section, we improve upon this result for the case when $\ell = 1$. The case of $\ell > 1$ is reserved for an extended version of the paper. Let $a_1, a_2 \in \mathbb{Z}_{n+2L+1}$ and $b \in \mathbb{F}_2$. Define $\mathbf{Y}_{T\wedge D}(n, a_1, a_2, b) \subseteq \mathbb{F}_2^n$ according to

$$\begin{aligned}\mathbf{Y}_{T\wedge D}(n,a_1,a_2,b) = \{\boldsymbol{x} \ &: \boldsymbol{x}' \in \mathcal{C}_{VT}(n,a_1,b,L), \\ &\sum_{i=1}^{n-1} (2i+1)^2 x_i \equiv a_2 \bmod (n+2L+1), \\ &x_n = 0\},\end{aligned}$$

where $L \geqslant 1$ is chosen so that $n+2L+1$ is a prime number greater than $4n-1$. Let $\mathcal{C}_{T\wedge D}(n,a_1,a_2,b) = \mathbf{Y}'_{T\wedge D}(n,a_1,a_2,b)$. We have the following lemma.

Lemma 13. *For all $a_1, a_2 \in \mathbb{Z}_{n+2L+1}$ and $b \in \mathbb{F}_2$, the code $\mathcal{C}_{T\wedge D}(n,a_1,a_2,b)$ is a 1-TD code.*

Proof: Since $\boldsymbol{x} \in \mathcal{C}_{VT}(n,a_1,b,L)$ by design, we only need to show that $\overline{\mathcal{C}}_{T\wedge D}(n,a_1,a_2,b) = \mathbf{Y}_{T\wedge D}(n,a_1,a_2,b)$ has Hamming distance 5. Note that if $\boldsymbol{x}' \in \mathcal{C}_{VT}(n,a_1,0,L)$ and $x_n = 0$, then this implies $\sum_{i=1}^{n-1}(2i+1)x_i \equiv a_1 \bmod n + 2L+1$. Thus, the vectors in the code $\mathcal{C}_{T\wedge D}(n,a_1,a_2,b)$ are a subset of the codewords in a Berlekamp code over $\mathbb{F}_{n+2L+1}$ which has minimum Lee distance 5 [12]. Therefore, since the code $\mathcal{C}_{T\wedge D}(n,a_1,a_2,b)$ is comprised of only binary vectors, the minimum Hamming distance of $\mathcal{C}_{T\wedge D}(n,a_1,a_2,b)$ is 5. ■

Corollary 14. *There exists a 1-TD code which redundancy at most $2\log(n) + c$ bits, for some absolute constant c.*

Hence, the above construction improves upon the general construction in (2) in terms of a saving of $\log(n)$ redundant bits.

V. Conclusion and Future Work

In this work, we considered the problem of coding for the Damerau distance. We focused on the problem of constructing codes capable of correcting a single deletion and a number of adjacent transpositions. Our extended work includes constructions of codes capable of correcting block adjacent transpositions and deletions.

Acknowledgment: This work was funded in part by the NSF grants CCF 11-17980, 1526875, 1527636. The work of Eitan Yaakobi was supported in part by the Israel Science Foundation (ISF) Grant No. 1624/14

References

[1] J. Brakensiek, V. Guruswami, and S. Zbarsky, "Efficient low-redundancy codes for correcting multiple deletions," *arXiv preprint arXiv:1507.06175*, 2015.

[2] E. Brill and R. C. Moore, "An improved error model for noisy channel spelling correction," in *Proceedings of the 38th Annual Meeting on Association for Computational Linguistics*. Association for Computational Linguistics, 2000, pp. 286–293.

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

[4] F. J. Damerau, "A technique for computer detection and correction of spelling errors," *Commun. ACM*, vol. 7, no. 3, pp. 171–176, Mar. 1964. [Online]. Available: http://doi.acm.org/10.1145/363958.363994

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

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

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

[8] S. Kumar, K. Tamura, and M. Nei, "Mega3: integrated software for molecular evolutionary genetics analysis and sequence alignment," *Briefings in bioinformatics*, vol. 5, no. 2, pp. 150–163, 2004.

[9] T. A. Le and H. D. Nguyen, "New multiple insertion-deletion correcting codes for non-binary alphabets," *arXiv preprint arXiv:1502.02727*, 2015.

[10] V. I. Levenshtein, "Binary codes capable of correcting deletions, insertions, and reversals," in *Soviet physics doklady*, vol. 10, no. 8, 1966, pp. 707–710.

[11] F. Paluncic, T. G. Swart, J. H. Weber, H. C. Ferreira, and W. A. Clarke, "A note on non-binary multiple insertion/deletion correcting codes," in *2011 IEEE Information Theory Workshop*.

[12] R. Roth, *Introduction to Coding Theory*. New York, NY, USA: Cambridge University Press, 2006.

[13] L. J. Schulman and D. Zuckerman, "Asymptotically good codes correcting insertions, deletions, and transpositions," *IEEE transactions on information theory*, vol. 45, no. 7, pp. 2552–2557, 1999.

[14] N. J. Sloane, "On single-deletion-correcting codes," *Codes and Designs, de Gruyter, Berlin*, pp. 273–291, 2002.

[15] R. Varshamov and G. Tenenholtz, "A code for correcting a single asymmetric error," *Automatica i Telemekhanika*, vol. 26, no. 2, pp. 288–292, 1965.

[16] M. M. Vilenchik and A. G. Knudson, "Endogenous dna double-strand breaks: production, fidelity of repair, and induction of cancer," *Proceedings of the National Academy of Sciences*, vol. 100, no. 22, pp. 12 871–12 876, 2003.

[17] S. Yazdi, H. M. Kiah, E. R. Garcia, J. Ma, H. Zhao, and O. Milenkovic, "Dna-based storage: Trends and methods," *arXiv preprint arXiv:1507.01611*, 2015.

[18] ——, "A rewritable, random-access dna-based storage system," *Nature Scientific Reports, http://www.nature.com/articles/srep14138*, 2015.