The Intersection of Insertion and Deletion Balls

Daniella Bar-Lev[*], **Omer Sabary**[†], **Yotam Gershon**[‡], and **Eitan Yaakobi**[*]

[*] Department of Computer Science, Technion — Israel Institute of Technology, Haifa, 3200003 Israel

[†] Department of Electrical and Computer Engineering, University of California San Diego, La Jolla, CA 92093, USA

[‡] Department of Electrical Engineering, Technion — Israel Institute of Technology, Haifa, 3200003 Israel

Emails: {daniellalev,yaakobi}@cs.technion.ac.il, osabary@ucsd.edu, yotamgr@campus.technion.ac.il

***Abstract*—This paper studies the intersections of insertion and deletion balls. The t-insertion, t-deletion ball of a sequence $\boldsymbol{x}$ is the set of all sequences received by t insertions, deletions to $\boldsymbol{x}$, respectively. While the intersection of either deletion balls or insertion balls has been rigorously studied before, the intersection of an insertion ball and a deletion ball has not been addressed so far. We find the maximum intersection size of any two insertion and deletion balls in the binary case. For the special case of one-insertion and one-deletion balls we find the intersection size for all pair of sequences. Then, we derive the largest and average values of this intersection size. Lastly, we present an algorithm that efficiently computes the intersection of any t_1-insertion ball and t_2-deletion ball.**

I. Introduction

Studying the size of the deletion and insertion balls as well as their intersections is the one of the more intriguing combinatorial problems in the area of coding for synchronization channels. The radius-t deletion ball of some sequence $\boldsymbol{x}$ is defined to be the set of all sequences that can be derived from $\boldsymbol{x}$ by exactly t deletions. Similarly the radius-t insertion ball of $\boldsymbol{x}$ is the set of all words that can be received by t insertions to $\boldsymbol{x}$. While it is well known that the insertion balls are regular, that is, the ball size does not depend on the center word $\boldsymbol{x}$, the deletion balls sizes indeed depend on the center $\boldsymbol{x}$ [10]. In particular, the minimum size of the deletion ball is achieved when $\boldsymbol{x}$ consists of a single symbol, while the maximum size is received only for the alternating words.

Studying [4], [12], [17] the intersection of deletion and insertion balls has several applications. For example, the largest size of these intersections provide the solution for the sequence reconstruction problem, which was first studied by Levenshtein [11], [12], for insertion and deletion channels. These largest sizes correspond to the required minimum number of channels when transmitting a codeword over several deletion or insertion channels. These problems have also connection to the generalized Gilbert-Varshamov bound [8], associative memories [9], [19], and list decoding [5], [13], [18]. One of the motivations to specifically study the intersection of a deletion ball together with an insertion ball originates from a recent problem we addressed in [14]. Assume a sequence $\boldsymbol{x}$ is transmitted over two channels, where the first introduces deletions while the second only insertions. In order to find the list of all possible transmitted sequences, it is necessary to find the intersection of the insertion ball of the first channel's output and the deletion ball of the second channel's output. While significant progress has been accomplished in studying the intersections of either deletion balls or insertion balls, to the best of our knowledge there is no study that consider together the intersection of insertion and deletion balls, which is the goal of this paper. Lastly, we note that studying the intersection of insertion and deletion balls contributes also to studying the balls in the Levenshtein metric and the intersection of these balls [2], [15], [16].

The rest of the paper is organized as follows. Section II presents the definitions used throughout the paper, a necessary and sufficient condition for the intersection of an insertion and a deletion to be nonempty, and the problems that will be solved in the paper. In Section III we study the maximum size of a t_1-insertion ball and a t_2-deletion ball for the binary case. Section IV addresses the case of one-insertion and one-deletion balls. We find the intersection size for all words $\boldsymbol{y}_1$ and $\boldsymbol{y}_2$ such that $\boldsymbol{y}_1$ is a subsequence of $\boldsymbol{y}_2$ for non-binary case. Based on this result we also derive the largest intersection size and the average size of this intersection. Lastly, in Section V we present an efficient algorithm to compute the intersection of two insertion and deletion balls. It is shown how to improve upon a naive solution computes first the deletion and insertion balls of the two sequences and then find their intersection. Due to the lack of space, some of the proofs in the paper are omitted.

II. Preliminaries and Problem Statement

Let Σ_q denote the set of integers $\{0, 1, \ldots, q-1\}$ and for an integer $n \geq 0$, let Σ_q^n be the set of all sequences (words) of length n over the alphabet Σ_q. For an integer t, $0 \leq t \leq n$, a sequence $\boldsymbol{y} \in \Sigma_q^{n-t}$ is a *t-subsequence* of $\boldsymbol{x} \in \Sigma_q^n$ if $\boldsymbol{y}$ can be obtained from $\boldsymbol{x}$ by deleting t symbols from $\boldsymbol{x}$. That is, there exist $n-t$ indices $1 \leq i_1 < i_2 < \cdots < i_{n-t} \leq n$ such that $y_j = x_{i_j}$, for all $1 \leq j \leq n-t$. We say that $\boldsymbol{y}$ is a *subsequence* of $\boldsymbol{x}$ if $\boldsymbol{y}$ is a t-subsequence of $\boldsymbol{x}$ for some t. Similarly, a sequence $\boldsymbol{y} \in \Sigma_q^{n+t}$ is a *t-supersequence* of $\boldsymbol{x} \in \Sigma_q^n$ if $\boldsymbol{x}$ is a t-subsequence of $\boldsymbol{y}$.

For a sequence $\boldsymbol{x} \in \Sigma_q^n$, let $\boldsymbol{x}_{[i,j]}$ be the subsequence $x_i x_{i+1} \cdots x_j$ and for a set of indices $U \subseteq \{1, \ldots, n\}$, the sequence $\boldsymbol{x}_U$ is the *projection* of $\boldsymbol{x}$ on the indices of U, which is the subsequence of $\boldsymbol{x}$ received by the symbols in the entries of U. For $\boldsymbol{x}, \boldsymbol{y} \in \Sigma_q^*$, the *Levenshtein distance* between $\boldsymbol{x}$ and $\boldsymbol{y}$, $d_L(\boldsymbol{x}, \boldsymbol{y})$, is the smallest number of insertions and deletions that is required to transform $\boldsymbol{x}$ into $\boldsymbol{y}$.

Definition 1. *The* **t-deletion ball** *centred at $\boldsymbol{x} \in \Sigma_q^n$, $D_t(\boldsymbol{x}) \subseteq \Sigma_q^{n-t}$, is the set of all t-subsequences of $\boldsymbol{x}$. The* **t-insertion ball** *centred at $\boldsymbol{x} \in \Sigma_q^n$, $I_t(\boldsymbol{x}) \subseteq \Sigma_q^{n+t}$, is the set of all t-supersequences of $\boldsymbol{x}$.*

For a sequence $\boldsymbol{x} \in \Sigma_q^n$, a *run* of $\boldsymbol{x}$ is a maximal subsequence $\boldsymbol{x}_{[i,j]}$ of identical symbols. The number of runs in $\boldsymbol{x}$ is denoted by $\rho(\boldsymbol{x})$. We say that $\boldsymbol{x}_{[i,j]}$ is an *alternating segment* if $\boldsymbol{x}_{[i,j]}$ is a sequence of alternating distinct symbols $\sigma, \sigma' \in \Sigma_q$. Note that $\boldsymbol{x}_{[i,j]}$ is a *maximal alternating segment* if $\boldsymbol{x}_{[i,j]}$ is an alternating segment and $\boldsymbol{x}_{[i-1,j]}, \boldsymbol{x}_{[i,j+1]}$ are not. For example, for $\boldsymbol{x} = 00110121$, $\rho(\boldsymbol{x}) = 5$ and the four maximal alternating segments are 0, 01, 101, 121.

In this work we study *the insertion-deletion intersection problem*. In this problem we let $\boldsymbol{y}_1, \boldsymbol{y}_2 \in \Sigma_q^*$ and $n \in \mathbb{N}$ such that $|\boldsymbol{y}_1| \leq n \leq |\boldsymbol{y}_2|$ and the goal is to find the set of all words $\boldsymbol{x} \in \Sigma_q^n$ such that $\boldsymbol{x}$ is both, a supersequence of $\boldsymbol{y}_1$ and a subsequences of $\boldsymbol{y}_2$. That is, to find the set

$$\mathsf{ID}(\boldsymbol{y}_1, \boldsymbol{y}_2, n) \triangleq I_{n-|\boldsymbol{y}_1|}(\boldsymbol{y}_1) \cap D_{|\boldsymbol{y}_2|-n}(\boldsymbol{y}_2).$$

The following lemma states a necessary and sufficient condition for $\mathsf{ID}(\boldsymbol{y}_1, \boldsymbol{y}_2, n) = \emptyset$.

Lemma 1. *Let $\boldsymbol{y}_1, \boldsymbol{y}_2 \in \Sigma_q^*$, and let n be an integer such that $|\boldsymbol{y}_1| \leq n \leq |\boldsymbol{y}_2|$. Then, $\mathsf{ID}(\boldsymbol{y}_1, \boldsymbol{y}_2, n) \neq \emptyset$ if and only if $\boldsymbol{y}_1$ is a subsequence of $\boldsymbol{y}_2$.*

Proof: Let $\delta \triangleq |\boldsymbol{y}_2| - |\boldsymbol{y}_1|$. Since $|\boldsymbol{y}_1| \leq |\boldsymbol{y}_2|$, we must perform at least δ insertions in order to transform $\boldsymbol{y}_1$ into $\boldsymbol{y}_2$, and hence $d_L(\boldsymbol{y}_1, \boldsymbol{y}_2) \geq \delta$. Assume $d_L(\boldsymbol{y}_1, \boldsymbol{y}_2) = \delta$ then $\boldsymbol{y}_1$ is obtained by deleting δ symbols from $\boldsymbol{y}_2$. That is, there exists a set of indices $U \subseteq [|\boldsymbol{y}_2|]$ such that $|U| = \delta$ and $(\boldsymbol{y}_2)_{[|\boldsymbol{y}_2|]\setminus U} = \boldsymbol{y}_1$. Let $U' \subseteq U$ be a set of indices such that $|U'| = |\boldsymbol{y}_2| - n$. It can be easily verified that

$$(\boldsymbol{y}_2)_{[|\boldsymbol{y}_2|]\setminus U'} \in \mathsf{ID}(\boldsymbol{y}_1, \boldsymbol{y}_2, n)$$

and hence, $\mathsf{ID}(\boldsymbol{y}_1, \boldsymbol{y}_2, n) \neq \emptyset$.

For the other direction, assume $d_L(\boldsymbol{y}_1, \boldsymbol{y}_2) > \delta$ and assume to the contrary that there exists $\boldsymbol{x} \in \Sigma_q^n$ such that $\boldsymbol{x} \in \mathsf{ID}(\boldsymbol{y}_1, \boldsymbol{y}_2, n)$. In this case, $\boldsymbol{x}$ can be obtained from $\boldsymbol{y}_2$ by $|\boldsymbol{y}_2| - n$ deletions and $\boldsymbol{y}_1$ can be obtained from $\boldsymbol{x}$ by $n - |\boldsymbol{y}_1|$ deletions, which is a contradiction since $d_L(\boldsymbol{y}_1, \boldsymbol{y}_2) > \delta$. ∎

According to Lemma 1, it is assumed in the rest of the paper that $\boldsymbol{y}_1$ is a subsequence of $\boldsymbol{y}_2$. The goal of this paper is to study the following problems.

Problem 1. *Given integers $n_1 \leq n \leq n_2$, find the maximum intersection size, i.e., $\max_{\boldsymbol{y}_1 \in \Sigma_q^{n_1}, \boldsymbol{y}_2 \in \Sigma_q^{n_2}} |\mathsf{ID}(\boldsymbol{y}_1, \boldsymbol{y}_2, n)|$.*

Problem 2. *Given integers $n_1 \leq n \leq n_2$, find the expected intersection size for only nonempty intersections,*

$$\mathop{\mathbb{E}}_{\substack{\boldsymbol{y}_1 \in \Sigma_q^{n_1}, \boldsymbol{y}_2 \in \Sigma_q^{n_2} \\ \boldsymbol{y}_1 \text{ is a subsequence of } \boldsymbol{y}_2}} \left[|\mathsf{ID}(\boldsymbol{y}_1, \boldsymbol{y}_2, n)|\right].$$

Problem 3. *Given $\boldsymbol{y}_1, \boldsymbol{y}_2 \in \Sigma_q^*$ and an integer n such that $|\boldsymbol{y}_1| \leq n \leq |\boldsymbol{y}_2|$, find the size of $\mathsf{ID}(\boldsymbol{y}_1, \boldsymbol{y}_2, n)$.*

Problem 4. *Given $\boldsymbol{y}_1, \boldsymbol{y}_2 \in \Sigma_q^*$ and an integer n such that $|\boldsymbol{y}_1| \leq n \leq |\boldsymbol{y}_2|$, find efficient algorithms to calculate the set $\mathsf{ID}(\boldsymbol{y}_1, \boldsymbol{y}_2, n)$.*

III. Maximal Intersection of Binary Insertion and Deletion Balls

In this section, we fully solve Problem 1 for the binary case, i.e., the case where $\boldsymbol{y}_1 \in \Sigma_2^{n_1}$ and $\boldsymbol{y}_2 \in \Sigma_2^{n_2}$.

Theorem 1. *For integers $0 \leq t_1 \leq n$, $0 \leq t_2$, and $q = 2$ we have that*

$$\max_{\boldsymbol{y}_1 \in \Sigma_2^{n-t_1}, \boldsymbol{y}_2 \in \Sigma_2^{n+t_2}} |\mathsf{ID}(\boldsymbol{y}_1, \boldsymbol{y}_2, n)| = \sum_{i=0}^{\min\{t_1, t_2\}} \binom{n}{i}.$$

Proof: Let $\boldsymbol{y}_1 \in \Sigma_q^{n-t_1}, \boldsymbol{y}_2 \in \Sigma_q^{n+t_2}$. If $t_1 \geq t_2$, we have that

$$|\mathsf{ID}(\boldsymbol{y}_1, \boldsymbol{y}_2, n)| = |I_{t_1}(\boldsymbol{y}_1) \cap D_{t_2}(\boldsymbol{y}_2)| \leq |D_{t_2}(\boldsymbol{y}_2)| \leq \sum_{i=0}^{t_2} \binom{n}{i},$$

where the last inequality is proven in [10]. To see that the upper bound is tight, let $\boldsymbol{y}_2$ be an alternating segment of length $n + t_2$ and let $\boldsymbol{y}_1$ be the word that is obtained by deleting the last $t_1 + t_2$ symbols of $\boldsymbol{y}_2$. Note that $\boldsymbol{y}_1$ is an alternating segment of length $n - t_1$. By [6], $|D_{t_2}(\boldsymbol{y}_2)| = \sum_{i=0}^{t_2} \binom{n}{i}$. In addition, any deletion in any word can decrease the number of runs by at most 2. Hence, for each $\boldsymbol{x} \in D_{t_2}(\boldsymbol{y}_2)$, $\rho(\boldsymbol{x}) \geq n + t_2 - (t_1 + t_2) = n - t_1$. In this case it is possible to show that we can delete t_1 more bits in $\boldsymbol{x}$ (one has to distinguish between the cases whether $\boldsymbol{x}$ and $\boldsymbol{y}_1$ start with the same bit) in order to receive $\boldsymbol{y}_1$, that is, $\boldsymbol{x}$ is a supersequence of $\boldsymbol{y}_1$ and hence $\boldsymbol{x} \in ID(\boldsymbol{y}_1, \boldsymbol{y}_2, n)$, which implies that $|\mathsf{ID}(\boldsymbol{y}_1, \boldsymbol{y}_2, n)| = \sum_{i=0}^{t_2} \binom{n}{i}$.

The case $t_1 < t_2$ is proved in a similar way using the insertion ball of $\boldsymbol{y}_1$ in order to upper bound the size of $|\mathsf{ID}(\boldsymbol{y}_1, \boldsymbol{y}_2, n)|$. ∎

IV. The Intersection of 1-Deletion Ball and 1-Insertion Ball

This section is focused on Problem 3 and presents an explicit characterization of the intersection of 1-deletion and 1-insertion balls. Using these results, we solve Problems 1 and 4 for the case where the balls' radius is one.

Since this section is focused on the specific case where $\boldsymbol{y}_1 \in \Sigma_q^{n-1}, \boldsymbol{y}_2 \in \Sigma_q^{n+1}$, and $d_L(\boldsymbol{y}_1, \boldsymbol{y}_2) = 2$, it holds that $\boldsymbol{y}_1$ can be obtained from $\boldsymbol{y}_2$ by exactly 2 deletions. The following lemma states the possible options to receive $\boldsymbol{y}_1$ by deletions from $\boldsymbol{y}_2$.

Lemma 2. *$\boldsymbol{y}_1$ can be obtained from $\boldsymbol{y}_2$ either by deleting two symbols from the same run, or by deleting them from two distinct runs, but not both.*

The next definition will be used in characterizing the intersection size $\mathsf{ID}(\boldsymbol{y}_1, \boldsymbol{y}_2, n)$.

Definition 2. *Let $\mathcal{R} : \Sigma_q^{n-1} \times \Sigma_q^{n+1} \to \{0, 1, 2\}$ be defined as follows,*

$$\mathcal{R}(\boldsymbol{y}_1, \boldsymbol{y}_2) \triangleq \begin{cases} 0 & d_L(\boldsymbol{y}_1, \boldsymbol{y}_2) > 2 \\ 1 & d_L(\boldsymbol{y}_1, \boldsymbol{y}_2) = 2 \text{ and } \boldsymbol{y}_2 \xrightarrow{1} \boldsymbol{y}_1 \\ 2 & d_L(\boldsymbol{y}_1, \boldsymbol{y}_2) = 2 \text{ and } \boldsymbol{y}_2 \xrightarrow{2} \boldsymbol{y}_1, \end{cases}$$

where $\boldsymbol{y}_2 \xrightarrow{i} \boldsymbol{y}_1$ denotes the case where $\boldsymbol{y}_1$ can be obtained from $\boldsymbol{y}_2$ by deletion(s) from i run(s).

Lemma 1 states that if $\mathcal{R}(\boldsymbol{y}_1, \boldsymbol{y}_2) = 0$, then $\mathsf{ID}(\boldsymbol{y}_1, \boldsymbol{y}_2, n) = \emptyset$. The size of $\mathsf{ID}(\boldsymbol{y}_1, \boldsymbol{y}_2, n)$ when $\mathcal{R}(\boldsymbol{y}_1, \boldsymbol{y}_2) = 1$ is given in the following lemma.

Lemma 3. *For $\boldsymbol{y}_1 \in \Sigma_q^{n-1}, \boldsymbol{y}_2 \in \Sigma_q^{n+1}$, if $\mathcal{R}(\boldsymbol{y}_1, \boldsymbol{y}_2) = 1$ then $\mathsf{ID}(\boldsymbol{y}_1, \boldsymbol{y}_2, n) = 1$.*

Proof: Since $\mathcal{R}(\boldsymbol{y}_1, \boldsymbol{y}_2) = 1$, there exists an index $1 \leq i \leq \rho(\boldsymbol{y}_2)$ such that two symbols can be deleted from the i-th run of $\boldsymbol{y}_2$ to obtain $\boldsymbol{y}_1$. It can be verified that the only word in $\mathsf{ID}(\boldsymbol{y}_1, \boldsymbol{y}_2, n)$ is the word obtained from $\boldsymbol{y}_2$ by deleting one symbol from its i-th run. ∎

To analyze the case where $\mathcal{R}(\boldsymbol{y}_1, \boldsymbol{y}_2) = 2$ we need to consider whether $\boldsymbol{y}_1$ can be obtained from $\boldsymbol{y}_2$ by deleting two consecutive runs or not. This will be done using the next definition.

Definition 3. *Let $\mathcal{A} : \Sigma_q^{n-1} \times \Sigma_q^{n+1} \to \{0, 1, \ldots, n-1\}$ be defined as follows. If $\boldsymbol{y}_1$ can be obtained from $\boldsymbol{y}_2$ by deleting two consecutive symbols and shortening a maximal alternating segment of m symbols in $\boldsymbol{y}_2$ to a maximal alternating segment of $m-2$ symbols in $\boldsymbol{y}_1$ then $\mathcal{A}(\boldsymbol{y}_1, \boldsymbol{y}_2) = m-2$, and $\mathcal{A}(\boldsymbol{y}_1, \boldsymbol{y}_2) = 0$ otherwise.*

For example, let $\boldsymbol{y}_1 = 0100\mathbf{0101}1$ and $\boldsymbol{y}_2 = 0100\mathbf{0101}011$. $\boldsymbol{y}_1$ can be obtained from $\boldsymbol{y}_2$ by deleting two consecutive symbols in the highlighted maximal alternating segment and hence $\mathcal{A}(\boldsymbol{y}_1, \boldsymbol{y}_2) = 6 - 2 = 4$.

Lemma 4. *Let $\boldsymbol{y}_1 \in \Sigma_q^{n-1}$ and $\boldsymbol{y}_2 \in \Sigma_q^{n+1}$ be two words. If $|\mathsf{ID}(\boldsymbol{y}_1, \boldsymbol{y}_2, n)| > 2$ then $\mathcal{A}(\boldsymbol{y}_1, \boldsymbol{y}_2) > 0$.*

Proof: Let $\boldsymbol{y}_1, \boldsymbol{y}_2$ be two words such that $|\mathsf{ID}(\boldsymbol{y}_1, \boldsymbol{y}_2, n)| > 2$. Lemma 3 implies that $\mathcal{R}(\boldsymbol{y}_1, \boldsymbol{y}_2) = 2$ and hence, there exist two indices $i < j$ such that $\boldsymbol{y}_1$ is obtained from $\boldsymbol{y}_2$ by deleting one symbol from the i-th run and one symbol from the j-th run of $\boldsymbol{y}_2$. Let $D_1(\boldsymbol{y}_2) = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_{\rho(\boldsymbol{y}_2)}\}$, where $\boldsymbol{x}_k$ denotes the

word obtained by deleting a symbol from the k-th run of $\boldsymbol{y}_2$. It is clear that $\boldsymbol{x}_i, \boldsymbol{x}_j$ are always supersequences of $\boldsymbol{y}_1$, since $\boldsymbol{x}_j$ is obtained by lengthening the run in $\boldsymbol{y}_1$ corresponding to the i-th run in $\boldsymbol{y}_2$, and vice versa. Since $|\mathsf{ID}(\boldsymbol{y}_1, \boldsymbol{y}_2, n)| > 2$, there exists an index $\ell \neq i, j$ such that $\boldsymbol{x}_\ell$ is a supersequence of $\boldsymbol{y}_1$. For $1 \leq k \leq \rho(\boldsymbol{y}_2)$, let σ_k be the symbol of the k-th run of $\boldsymbol{y}_2$ and r_k is its length, i.e.,

$$\boldsymbol{y}_2 = \sigma_1^{r_1} \cdots \sigma_{i-1}^{r_{i-1}} \sigma_i^{r_i} \sigma_{i+1}^{r_{i+1}} \cdots \sigma_{j-1}^{r_{j-1}} \sigma_j^{r_j} \sigma_{j+1}^{r_{j+1}} \cdots \sigma_{\rho(\boldsymbol{y}_2)}^{r_{\rho(\boldsymbol{y}_2)}},$$
$$\boldsymbol{y}_1 = \sigma_1^{r_1} \cdots \sigma_{i-1}^{r_{i-1}} \sigma_i^{r_i-1} \sigma_{i+1}^{r_{i+1}} \cdots \sigma_{j-1}^{r_{j-1}} \sigma_j^{r_j-1} \sigma_{j+1}^{r_{j+1}} \cdots \sigma_{\rho(\boldsymbol{y}_2)}^{r_{\rho(\boldsymbol{y}_2)}},$$

where σ^0 is defined to be the empty word. First, we prove that $\ell < i$. Assume to the contrary that $i < \ell$, we have that,

$$\boldsymbol{x}_\ell = \sigma_1^{r_1} \cdots \sigma_{i-1}^{r_{i-1}} \sigma_i^{r_i} \cdots \sigma_\ell^{r_\ell - 1} \cdots \sigma_{\rho(\boldsymbol{y}_2)}^{r_{\rho(\boldsymbol{y}_2)}}.$$

If $r_i > 1$ or $\sigma_{i-1} \neq \sigma_{i+1}$ then $\boldsymbol{y}_1, \boldsymbol{x}_\ell$ are equal up to the $(i-1)$-th run and the i-th run of $\boldsymbol{x}_\ell$ is longer than the i-th run of $\boldsymbol{y}_1$ (if exists) by one. Hence, $\boldsymbol{y}_1$ must be obtained from $\boldsymbol{x}_\ell$ by deleting a symbol from the i-th run. Otherwise, we have that $r_i = 1$, $\sigma_{i-1} = \sigma_{i+1}$ and $\boldsymbol{y}_1, \boldsymbol{x}_\ell$ are equal up to the $(i-2)$-th run and the $(i-1)$-th run of $\boldsymbol{y}_1$ is longer than the $(i-1)$-th run of $\boldsymbol{x}_\ell$ by r_{i+1}. It can be verified that since $\boldsymbol{x}_\ell$ is a supersequence of $\boldsymbol{y}_1$, $\boldsymbol{y}_1$ must be obtained from $\boldsymbol{x}_\ell$ by deleting a symbol from the i-th run, as before. Hence, $\boldsymbol{y}_1$ can be obtained from $\boldsymbol{y}_2$ by deleting one symbol from the i-th run and one more symbol, either from the j-th run or from the ℓ-th run. Since $\boldsymbol{y}_1$ can be obtained from $\boldsymbol{x}_i$ by deleting a symbol from the j-th or the ℓ-th run (with respect to their indices in $\boldsymbol{y}_2$), it holds that the j-th and the ℓ-th run of $\boldsymbol{y}_2$ are combined to a single run in $\boldsymbol{x}_i$. This implies that either $j < i$ or $\ell < i$ which is a contradiction.

Next, since $\ell < i$ we have that

$$\boldsymbol{x}_\ell = \sigma_1^{r_1} \cdots \sigma_{\ell-1}^{r_{\ell-1}} \sigma_\ell^{r_\ell - 1} \cdots \sigma_i^{r_i} \cdots \sigma_{\rho(\boldsymbol{y}_2)}^{r_{\rho(\boldsymbol{y}_2)}},$$
$$\boldsymbol{y}_1 = \sigma_1^{r_1} \cdots \sigma_{\ell-1}^{r_{\ell-1}} \sigma_\ell^{r_\ell} \cdots \sigma_i^{r_i - 1} \sigma_{i+1}^{r_{i+1}} \cdots \sigma_{\rho(\boldsymbol{y}_2)}^{r_{\rho(\boldsymbol{y}_2)}}.$$

If $r_\ell > 1$ or $\sigma_{\ell-1} \neq \sigma_{\ell+1}$, then $\boldsymbol{y}_1, \boldsymbol{x}_\ell$ are equal up to the $(\ell-1)$-th run and the ℓ-th run of $\boldsymbol{y}_1$ is longer than the ℓ-th run of $\boldsymbol{x}_\ell$ (if exists) by one. Hence, $\boldsymbol{y}_1$ must be obtained from $\boldsymbol{x}_\ell$ by deleting a symbol from the $(\ell+1)$-th run, which must be of length one. Hence

$$\sigma_\ell \sigma_{\ell+1} \cdots \sigma_i^{r_i - 1} = \sigma_{\ell+2} \sigma_{\ell+3} \cdots \sigma_i^{r_i} \sigma_{i+1}$$

and it can be verified that the last symbol of the ℓ-th run and the first symbol of the i-th run are part of the same alternating segment of $\boldsymbol{y}_2$. If $i > \ell + 1$ then this is an alternating segment of length three or more and by definition $\mathcal{A}(\boldsymbol{y}_1, \boldsymbol{y}_2) > 0$. Otherwise, $i = \ell + 1$, and $\boldsymbol{y}_1$ is obtained from $\boldsymbol{x}_\ell$ by deleting one symbol from the i-th run and one symbol either from the j-th or from the ℓ-th run of $\boldsymbol{y}_2$. The latter can be hold only if $i = \ell + 1$, $j = \ell + 2$ and $\sigma_\ell = \sigma_j$ which implies that $\mathcal{A}(\boldsymbol{y}_1, \boldsymbol{y}_2) > 0$. Otherwise, we have that $r_\ell = 1$, $\sigma_{\ell-1} = \sigma_{\ell+1}$ and $\boldsymbol{y}_1, \boldsymbol{x}_\ell$ are equal up to the $(\ell-2)$-th run and the $(\ell-1)$-th run of $\boldsymbol{x}_\ell$ is longer than the $(\ell-1)$-th run of $\boldsymbol{y}_1$ by $r_{\ell+1}$. Since $\boldsymbol{x}_\ell$ is a supersequence of $\boldsymbol{y}_1$, $\boldsymbol{y}_1$ must be obtained from $\boldsymbol{x}_\ell$ by deleting a symbol from the $(\ell+1)$-th run, and by similar arguments, $\mathcal{A}(\boldsymbol{y}_1, \boldsymbol{y}_2) > 0$. ∎

The following lemma is an immediate result of Lemma 4.

Lemma 5. *Let $\boldsymbol{y}_1 \in \Sigma_q^{n-1}$ and $\boldsymbol{y}_2 \in \Sigma_q^{n+1}$ be two words. If $\mathcal{R}(\boldsymbol{y}_1, \boldsymbol{y}_2) = 2$ and $\mathcal{A}(\boldsymbol{y}_1, \boldsymbol{y}_2) = 0$ then $\mathsf{ID}(\boldsymbol{y}_1, \boldsymbol{y}_2, n) = 2$.*

The last case for $\boldsymbol{y}_1$ and $\boldsymbol{y}_2$ is handled in the next lemma.

Lemma 6. *Let $\boldsymbol{y}_1 \in \Sigma_q^{n-1}$ and $\boldsymbol{y}_2 \in \Sigma_q^{n+1}$ be two words. If $\mathcal{R}(\boldsymbol{y}_1, \boldsymbol{y}_2) = 2$ and $\mathcal{A}(\boldsymbol{y}_1, \boldsymbol{y}_2) = m - 2$ then $\mathsf{ID}(\boldsymbol{y}_1, \boldsymbol{y}_2, n) = m$.*

Proof: By definition, $\boldsymbol{y}_1$ can be obtained from $\boldsymbol{y}_2$ by deleting two consecutive symbols from a maximal alternating segment of length m. Denote by $j, j+1, \ldots, j+m-1$ the indices of the runs which correspond to this maximal alternating segment. Let $D_1(\boldsymbol{y}_2) = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_{\rho(\boldsymbol{y}_2)}\}$, where $\boldsymbol{x}_i$ is obtained from $\boldsymbol{y}_2$ by deleting one symbol from the i-th run of $\boldsymbol{y}_2$. It can be verified that for any $i \in \{j, \ldots, j+m-1\}$, $\boldsymbol{y}_1$ can be obtained from $\boldsymbol{x}_i$ by deleting a symbol from the $(i-1)$-th run or the $(i+1)$-th run of $\boldsymbol{x}_i$ (the one that is part of the maximal alternating segment). Hence $\boldsymbol{x}_i \in \mathsf{ID}(\boldsymbol{y}_1, \boldsymbol{y}_2, n)$. Otherwise, $i \notin \{j, \ldots, j+m-1\}$. Assume to the contrary that $\boldsymbol{x}_i$ is a supersequence of $\boldsymbol{y}_1$. By similar arguments to those presented in the proof of Lemma 4, it can be concluded that one of the symbols in the i-th run of $\boldsymbol{y}_2$ is part of the same alternating segment as the j-th run, which is a contradiction. ∎

Theorem 2. *For any two words $\boldsymbol{y}_1 \in \Sigma_q^{n-1}$ and $\boldsymbol{y}_2 \in \Sigma_q^{n+1}$, we have that*

$$\mathsf{ID}(\boldsymbol{y}_1, \boldsymbol{y}_2, n) = \mathcal{R}(\boldsymbol{y}_1, \boldsymbol{y}_2) + \mathcal{A}(\boldsymbol{y}_1, \boldsymbol{y}_2).$$

Proof: If $d_L(\boldsymbol{y}_1, \boldsymbol{y}_2) > 2$ then by the definitions of $\mathcal{R}, \mathcal{A}$ we have that $\mathsf{ID}(\boldsymbol{y}_1, \boldsymbol{y}_2, n) = 0$. Otherwise, by Corollary 2 it holds that either $\boldsymbol{y}_2 \xrightarrow{1} \boldsymbol{y}_1$ or $\boldsymbol{y}_2 \xrightarrow{2} \boldsymbol{y}_1$. If $\boldsymbol{y}_2 \xrightarrow{1} \boldsymbol{y}_1$ then by Lemma 3, $\mathsf{ID}(\boldsymbol{y}_1, \boldsymbol{y}_2, n) = 1$ and by the definitions of $\mathcal{R}, \mathcal{A}$,

$$\mathcal{R}(\boldsymbol{y}_1, \boldsymbol{y}_2) + \mathcal{A}(\boldsymbol{y}_1, \boldsymbol{y}_2) = 1 + 0 = 1$$

Lastly, if $\boldsymbol{y}_2 \xrightarrow{2} \boldsymbol{y}_1$ then the result follows from Lemma 5 and Lemma 6. ∎

Corollary 1. *For any two integers $n, q > 1$ we have that*

$$\max_{\boldsymbol{y}_1 \in \Sigma_q^{n-1}, \boldsymbol{y}_2 \in \Sigma_q^{n+1}} |\mathsf{ID}(\boldsymbol{y}_1, \boldsymbol{y}_2, n)| = n + 1.$$

Note that the latter corollary is a generalization of Theorem 1 for any $q \geq 2$, where the balls' radius is one. Lastly, using Theorem 2, it is possible to calculate the expected size of the set $\mathsf{ID}(\boldsymbol{y}_1, \boldsymbol{y}_2, n)$. The result is given in the next theorem.

Theorem 3. *For any two integers $n, q > 1$ we have that*

$$\mathop{\mathbb{E}}_{\substack{\boldsymbol{y}_1 \in \Sigma_q^{n-1}, \boldsymbol{y}_2 \in \Sigma_q^{n+1} \\ \boldsymbol{y}_1 \text{ is a subsequence of } \boldsymbol{y}_2}} [|\mathsf{ID}(\boldsymbol{y}_1, \boldsymbol{y}_2, n)|] = 2 - \frac{q}{1 + (q-1)(n+1) + (q-1)^2 \binom{n+1}{2}}.$$

V. Efficient Algorithm Computing the Intersection of Insertion and Deletion Balls

In this section, we address Problem 4 and present an efficient algorithm that given $y_1 \in \Sigma_2^{n-t_1}$, $y_2 \in \Sigma_2^{n+t_2}$, and n calculates $|\mathsf{ID}(\boldsymbol{y}_1, \boldsymbol{y}_2, n)|$. A naive method to calculate this intersection is to compute these two balls, i.e., $I_{t_1}(\boldsymbol{y}_1)$ and $D_{t_2}(\boldsymbol{y}_2)$, and then calculate their intersection. However, since the calculation of the balls is done recursively, this approach will introduce high complexity[1]. The algorithm described in this section is based on dynamic programming, and hence works more efficiently.

The following additional definitions will be use throughout the section. A sequence $\boldsymbol{x}$ is called a *common subsequence* of some words $\boldsymbol{y}_1, \ldots, \boldsymbol{y}_t$ if $\boldsymbol{x}$ is a subsequence of each one of these t words. The set of all common subsequences of $\boldsymbol{y}_1, \ldots, \boldsymbol{y}_t$ is denoted by $\mathcal{CS}(\boldsymbol{y}_1, \ldots, \boldsymbol{y}_t)$ and $\mathsf{LCS}(\boldsymbol{y}_1, \ldots, \boldsymbol{y}_t)$ denotes the *length of the longest common subsequence* (*LCS*) of $\boldsymbol{y}_1, \ldots, \boldsymbol{y}_t$, that is, $\mathsf{LCS}(\boldsymbol{y}_1, \ldots, \boldsymbol{y}_t) = \max_{\boldsymbol{x} \in \mathcal{CS}(\boldsymbol{y}_1, \ldots, \boldsymbol{y}_t)} \{|\boldsymbol{x}|\}$. The set of

[1] since the size of $I_{n-|\boldsymbol{y}_1|}(\boldsymbol{y}_1)$ is $\Theta(|\boldsymbol{y}_1|^{n-|\boldsymbol{y}_1|})$ and the largest size of $D_{|\boldsymbol{y}_2|-n}(\boldsymbol{y}_2)$ is $\Theta(|\boldsymbol{y}_2|^{|\boldsymbol{y}_2|-n})$, the worst case complexity of this solution is $\Theta(n^{|\boldsymbol{y}_2|-|\boldsymbol{y}_1|+1}) = \Theta(n^{t_1+t_2+1})$.

all shortest common subsequences of $\boldsymbol{y}_1, \ldots, \boldsymbol{y}_t$ is denoted by $\mathcal{LCS}(\boldsymbol{y}_1, \ldots, \boldsymbol{y}_t)$.

Let us denote by $\mathcal{U}(\boldsymbol{y}_1, \boldsymbol{y}_2)$ the set of index sets U such that the projection of $\boldsymbol{y}_2$ on the index set U yields $\boldsymbol{y}_1$, that is,

$$\mathcal{U}(\boldsymbol{y}_1, \boldsymbol{y}_2) = \{U \subseteq [|\boldsymbol{y}_2|] : \ (\boldsymbol{y}_2)_U = \boldsymbol{y}_1\}.$$

We say that an index set U is a *right-most index set* of a sequence $\boldsymbol{x}$ if all the indices of U are the right most indices in their run in $\boldsymbol{y}$, i.e., for all $i \in U$, either i is the right most index of its run in $\boldsymbol{y}$, or $i+1 \in U$. Furthermore, denote by $\mathcal{U}_{right}(\boldsymbol{y}_1, \boldsymbol{y}_2) \subseteq \mathcal{U}(\boldsymbol{y}_1, \boldsymbol{y}_2)$ the set of all right-most index sets of $\boldsymbol{y}_2$ in $\mathcal{U}(\boldsymbol{y}_1, \boldsymbol{y}_2)$[2]. We next show how to exhaustively and efficiently scan all vectors in the set $\mathsf{ID}(\boldsymbol{y}_1, \boldsymbol{y}_2, n)$. This will be done by first considering the right-most index sets of $\boldsymbol{y}_2$ that their projection yields $\boldsymbol{y}_1$ and then complete them with an arbitrary set of indices on the remaining set of indices to receive a length-n word in $\mathsf{ID}(\boldsymbol{y}_1, \boldsymbol{y}_2, n)$.

Theorem 4. *It holds that*

$$\mathsf{ID}(\boldsymbol{y}_1, \boldsymbol{y}_2, n) = \{(\boldsymbol{y}_2)_{U \cup V} : U \in \mathcal{U}_{right}(\boldsymbol{y}_1, \boldsymbol{y}_2), V \subseteq [|\boldsymbol{y}_2|] \setminus U, \ |V| = n - |U|\}.$$

Proof: Let $U \in \mathcal{U}_{right}(\boldsymbol{y}_1, \boldsymbol{y}_2)$. By definition, $(\boldsymbol{y}_2)_U = \boldsymbol{y}_1$. Hence, $\boldsymbol{y}_1$ is a subsequence of $(\boldsymbol{y}_2)_{U \cup V}$. In addition, since $U \cup V \subseteq [|\boldsymbol{y}_2|]$, $\boldsymbol{y}_2$ is a supersequence of $(\boldsymbol{y}_2)_{U \cup V}$. That is, $\boldsymbol{y}_2$ can be obtained from $(\boldsymbol{y}_2)_{U \cup V}$ by $|\boldsymbol{y}_2| - |U \cap V|$ insertions and $\boldsymbol{y}_1$ can be obtained from $(\boldsymbol{y}_2)_{U \cup V}$ by $|V|$ deletions. Namely, that is to say that $(\boldsymbol{y}_2)_{U \cup V} \in \mathsf{ID}(\boldsymbol{y}_1, \boldsymbol{y}_2, n)$.

For the other direction, let $\boldsymbol{x} \in \mathsf{ID}(\boldsymbol{y}_1, \boldsymbol{y}_2, n)$ be a sequence. By definition, $\boldsymbol{x}$ is a subsequence of $\boldsymbol{y}_2$. Hence, there is a set of indices $T \subseteq [|\boldsymbol{y}_2|]$ such that $(\boldsymbol{y}_2)_T = \boldsymbol{x}$. Let the number of indices in T that are contained in the i-th run of $\boldsymbol{y}_2$ be denoted by $T(i)$ and let T' be the index set that consists of the $T(i)$ right-most indices of the i-th run of $\boldsymbol{y}_2$ for $1 \leq i \leq r(\boldsymbol{y}_2)$. Then. it holds that $(\boldsymbol{y}_2)_{T'} = \boldsymbol{x}$. In addition, since $\boldsymbol{y}_1$ is a subsequence of $\boldsymbol{x} = (\boldsymbol{y}_2)_{T'}$, there is a set of indices $U \subseteq T'$ such that $(\boldsymbol{y}_2)_U = \boldsymbol{y}_1$. Since the indices in T' are the right-most indices of each run of $\boldsymbol{y}_2$, there is an index set $U' \subseteq T'$ that consists of the $U(i)$ right-most indices of the i-th run of $\boldsymbol{y}_2$ for $1 \leq i \leq \rho(\boldsymbol{y}_2)$ and satisfies $(\boldsymbol{y}_2)_{U'} = \boldsymbol{y}_1$, where $\rho(\boldsymbol{y}_2)$ denotes the number of runs in $\boldsymbol{y}_2$. That is, $U' \in \mathcal{U}_{right}(\boldsymbol{y}_1, \boldsymbol{y}_2)$ and for $V = T' \setminus U'$ we have that $(\boldsymbol{y}_2)_{U' \cup V} = (\boldsymbol{y}_2)_{T'} = \boldsymbol{x}$. ∎

Using Theorem 4 we have the following algorithm for the calculation of the intersection $\mathsf{ID}(\boldsymbol{y}_1, \boldsymbol{y}_2, n)$,

Algorithm 1 $\mathsf{ID}(\boldsymbol{y}_1, \boldsymbol{y}_2, n)$

Input: $\boldsymbol{y}_1, \boldsymbol{y}_2, n$
Calculate $\mathcal{U}_{right}(\boldsymbol{y}_1, \boldsymbol{y}_2)$
Set $S = \emptyset$
for each $U \in \mathcal{U}_{right}(\boldsymbol{y}_1, \boldsymbol{y}_2)$ **do**
  for each $V \subseteq [|\boldsymbol{y}_2|] \setminus U$ such that $|V| = n - |U|$ **do**
    Calculate $\boldsymbol{x} = (\boldsymbol{y}_2)_{U \cup V}$
    Set $S = S \cup \{\boldsymbol{x}\}$
Return S

Notice that replacing any index in $j \in V$ with an index $j' \in [|\boldsymbol{y}_2|] \setminus (U \cup V)$ from the same run of $\boldsymbol{y}_2$ has no affect on $(\boldsymbol{y}_2)_{U \cup V}$. That is, $(\boldsymbol{y}_2)_{U \cup V} = (\boldsymbol{y}_2)_{U \cup (V \setminus \{j\}) \cup j'}$ for any two indices $j, j' \notin U$ such that $j \in V, j' \notin V$ and j, j' belong to the same run in $\boldsymbol{y}_2$. Hence, in the second for loop of Algorithm 1, it is sufficient to iterate only over indices sets V that differ in the number of indices from the same run.

It is left to calculate the sets of index sets $\mathcal{U}_{right}(\boldsymbol{y}_1, \boldsymbol{y}_2)$. The algorithm to calculate $\mathcal{U}_{right}(\boldsymbol{y}_1, \boldsymbol{y}_2)$ is based on the dynamic programming implementation of the LCS problem [7], which is shortly explained next. Given two words $\boldsymbol{x}, \boldsymbol{y}$, let $\mathsf{LCS}(i, j)$ denote the length of the LCS of $\boldsymbol{x}_{[i]}$ and $\boldsymbol{y}_{[j]}$. The length of the LCS of $\boldsymbol{x}$ and $\boldsymbol{y}$ is given by $\mathsf{LCS}(|\boldsymbol{x}|, |\boldsymbol{y}|)$ and is computed using the following recursive formula

$$\mathsf{LCS}(i, j) = \begin{cases} 0 & i = 0 \text{ or } j = 0 \\ 1 + \mathsf{LCS}(i-1, j-1) & \boldsymbol{x}(i) = \boldsymbol{y}(j) \\ \max\{\mathsf{LCS}(i-1, j), \mathsf{LCS}(i, j-1)\} & \text{otherwise} \end{cases}$$

The implementation of this computation is done using a $(|\boldsymbol{x}| + 1) \times (|\boldsymbol{y}| + 1)$ matrix, which is referred as the *dynamic programming table*, where the j-th entry of the i-th row of the dynamic programming table quals to $\mathsf{LCS}(i, j)$.

The calculation of the set of index sets in $\mathcal{U}_{right}(\boldsymbol{y}_1, \boldsymbol{y}_2)$ will also be done using the $(|\boldsymbol{y}_1| + 1) \times (|\boldsymbol{y}_2| + 1)$ dynamic programming table. Observe that under the problem specifications, $\boldsymbol{y}_1$ is a subsequence of $\boldsymbol{y}_2$, and thus it holds that $\mathcal{LCS}(\boldsymbol{y}_1, \boldsymbol{y}_2) = \{\boldsymbol{y}_1\}$ and thus $\mathsf{LCS}(|\boldsymbol{y}_1|, |\boldsymbol{y}_2|) = |\boldsymbol{y}_1|$. Therefore, there exists at least one index set U such that $(\boldsymbol{y}_2)_U = \boldsymbol{y}_1$. However, in order to find all such right-most index sets, the idea is to search within the dynamic programming table, in order to identify these index sets U which consist of only the right-most indices of each run of $\boldsymbol{y}_2$ and satisfy $(\boldsymbol{y}_2)_U = \boldsymbol{y}_1$.

In order to search such index sets U within the dynamic programming table in an efficient way, we use some of its properties. First, note that the size of each such set U is $|\boldsymbol{y}_1|$. Let $U = \{i_1, i_2, \ldots, i_{|\boldsymbol{y}_1|}\}$, since $(\boldsymbol{y}_2)_U = \boldsymbol{y}_1$, the j-th entry of the i_j-th column of the dynamic programming table equals to j. By the above, the column that corresponds to the i_j-th index contains the value j in the j-th entry, and the i_j-th index represents the appearance of the j-th symbol of $\boldsymbol{y}_1$ in $\boldsymbol{y}_2$. Since each selected index i_j corresponds to the j-th symbol of $\boldsymbol{y}_1$, which corresponds to the j-th row of the dynamic programming table, the selection of i_j allows us to eliminate the j-th row of the dynamic programming table from the rest of the search. We choose the indices of U in a backward order, and by doing so each selection of index reduces the number of rows by one in the matrix we need to search in the rest of the algorithm.

Before we present the explicit algorithm to compute the set $\mathcal{U}_{right}(\boldsymbol{y}_1, \boldsymbol{y}_2)$, let us introduce a few data structures that will be used in the algorithm.

1) A two-dimensional array, called match, such that $\mathsf{match}(i, j) = 1$ if $\boldsymbol{y}_1[i] = \boldsymbol{y}_2[j]$ and otherwise 0.
2) A two-dimensional array LCS, which is the dynamic programming table of the LCS algorithm.
3) A binary vector curr of length $|\boldsymbol{y}_2|$. The set of non-zero entries of curr correspond to the indices in a set $U \in \mathcal{U}_{right}(\boldsymbol{y}_1, \boldsymbol{y}_2)$ in the current step of the search.

By using the above characterization, we design the following recursive procedure, presented in Algorithm 2, in order to calculate the set $\mathcal{U}_{right}(\boldsymbol{y}_1, \boldsymbol{y}_2)$. Algorithm 2 is initially invoked with the sequences $\boldsymbol{y}_1$ and $\boldsymbol{y}_2$, the indices $i = |\boldsymbol{y}_1|, \ j = |\boldsymbol{y}_2|$, and the all zero vector curr.

The next example demonstrates the idea of Algorithm 2.

[2]Not to be confused with the right canonical embedding (also called canonical embedding) presented in [3]. The right canonical embedding is defined as the embedding that consists of the largest possible indices, and note that the canonical embedding is unique.

Algorithm 2 The calculation of $\mathcal{U}_{right}$

Input: $\boldsymbol{y}_1$, $\boldsymbol{y}_2$, i, j and a pointer to curr
$\mathcal{U}_{right} = \emptyset$.
if $i = 0$ or $j = 0$ **then**
  $\mathcal{U}_{right} \leftarrow \{\mathsf{curr}\} \cup \mathcal{U}_{right}$
  Return $\mathcal{U}_{right}$
if $\mathsf{LCS}(i, j) < i$ **then**
  Return $\mathcal{U}_{right}$
if $\mathsf{match}(i, j) = 1$ and ($j = n$ or $\boldsymbol{y}_2[j] \neq \boldsymbol{y}_2[j-1]$ or $\mathsf{curr}[j] = 1$) **then**
  Set $\mathsf{curr}[j-1] \leftarrow 1$
  $\mathcal{U}_{right} \leftarrow \mathcal{U}_{right} \cup$ Algorithm 2$(\mathsf{curr}, \boldsymbol{y}_1, \boldsymbol{y}_2, i-1, j-1)$
  Set $\mathsf{curr}[j-1] \leftarrow 0$
if $\mathsf{LCS}(i, j-1) = i$ **then**
  $\mathcal{U}_{right} \leftarrow \mathcal{U}_{right} \cup$ Algorithm 2$(\mathsf{curr}, \boldsymbol{y}_1, \boldsymbol{y}_2, i, j-1)$
Return $\mathcal{U}_{right}$.

Example 1. *Consider the example shown in Fig. 1, in which* $\boldsymbol{y}_1 = 0010$ *and* $\boldsymbol{y}_2 = 000111010$ *and each table is the dynamic programming table. The four highlighted columns in each table correspond to one set of indices* $U \in \mathcal{U}_{right}$.

		0	0	0	1	1	1	0	1	0
	0	0	0	0	0	0	0	0	0	0
0	0	1	1	1	1	1	1	1	1	1
0	0	1	2	2	2	2	2	2	2	2
1	0	1	2	2	3	3	3	3	3	3
0	0	1	2	2	3	3	3	4	4	4

		0	0	0	1	1	1	0	1	0
	0	0	0	0	0	0	0	0	0	0
0	0	1	1	1	1	1	1	1	1	1
0	0	1	2	2	2	2	2	2	2	2
1	0	1	2	2	3	3	3	3	3	3
0	0	1	2	2	3	3	3	4	4	4

		0	0	0	1	1	1	0	1	0
	0	0	0	0	0	0	0	0	0	0
0	0	1	1	1	1	1	1	1	1	1
0	0	1	2	2	2	2	2	2	2	2
1	0	1	2	2	3	3	3	3	3	3
0	0	1	2	2	3	3	3	4	4	4

		0	0	0	1	1	1	0	1	0
	0	0	0	0	0	0	0	0	0	0
0	0	1	1	1	1	1	1	1	1	1
0	0	1	2	2	2	2	2	2	2	2
1	0	1	2	2	3	3	3	3	3	3
0	0	1	2	2	3	3	3	4	4	4

Fig. 1: An example of Algorithm 2 for $\boldsymbol{y}_1 = 0010$ and $\boldsymbol{y}_2 = 000111010$, so $\mathsf{LCS}(\boldsymbol{y}_1, \boldsymbol{y}_2) = 4$. The highlighted columns represent the right-most index sets $U \in \mathcal{U}_{right}(\boldsymbol{y}_1, \boldsymbol{y}_2)$. More specifically, theses sets, ordered in a clockwise manner, are: $\{3, 7, 8, 9\}, \{2, 3, 6, 9\}, \{2, 3, 8, 9\}, \{2, 3, 6, 7\}$.

Theorem 5. *Let* $\boldsymbol{y}_1$ *and* $\boldsymbol{y}_2$ *be two sequences such that* $|\boldsymbol{y}_1| \leq |\boldsymbol{y}_2|$. *The output of Algorithm 2 is* $\mathcal{U}_{right}(\boldsymbol{y}_1, \boldsymbol{y}_2)$ *if* $\boldsymbol{y}_1$ *is a subsequence of* $\boldsymbol{y}_2$, *and is* $\emptyset$ *otherwise.*

Proof: Let $\mathcal{U}_{right}$ denote the output of Algorithm 2 for an input of $\boldsymbol{y}_1$ and $\boldsymbol{y}_2$, i.e.,

$$\mathcal{U}_{right} = \text{Algorithm 2}(\mathsf{curr}, \boldsymbol{y}_1, \boldsymbol{y}_2, |\boldsymbol{y}_1|, |\boldsymbol{y}_2|).$$

If $\boldsymbol{y}_1$ is not a subsequence of $\boldsymbol{y}_2$ it must be that $1 \leq |\boldsymbol{y}_1|$, and by the assumption, $|\boldsymbol{y}_1| \leq |\boldsymbol{y}_2|$. Thus, the first if condition does not hold. Also, it must be that $\mathsf{LCS}(|\boldsymbol{y}_1|, |\boldsymbol{y}_2|) < |\boldsymbol{y}_1|$, so the algorithm output is $\mathcal{U}_{right} = \emptyset$.

Let us assume that $\boldsymbol{y}_1$ is a subsequence of $\boldsymbol{y}_2$. First, we will prove that $\mathcal{U}_{right} \subseteq \mathcal{U}_{right}(\boldsymbol{y}_1, \boldsymbol{y}_2)$. At the initialization $i \leq j$ and in each recursive call we can decrease both i and j by one or only decrease j by one. Since the latter is possible only if $i = \mathsf{LCS}(i, j-1) \leq \min\{i, j-1\} \leq j-1$, it holds that $i \leq j$ throughout the run of the algorithm. For convenience, we consider the vector curr to be the index set $U := \{j \mid \mathsf{curr}[j-1] = 1\}$ it represents. An index set U is inserted to $\mathcal{U}_{right}$ only when the first if condition holds, that is, only if $i = 0$ or $j = 0$ and since $i \leq j$, that is, whenever $i = 0$. By the definition of the algorithm, i and U can only be changed when the third if condition holds. In this case, j is inserted to U and i and j are decreased by one. Thus, if $i = 0$, U must contain $|\boldsymbol{y}_1|$ indices $j_1, \ldots, j_{|\boldsymbol{y}_1|}$ and by the third condition each j_i satisfies $(\boldsymbol{y}_1)_i = (\boldsymbol{y}_2)_{j_i}$. Therefore, each index set U in the output satisfies $\boldsymbol{y}_1 = (\boldsymbol{y}_2)_U$ which implies that $\mathcal{U}_{right} \subseteq \mathcal{U}(\boldsymbol{y}_1, \boldsymbol{y}_2)$.

Let $U \in \mathcal{U}_{rignt}$, and denote by $j_1 < j_2 < \cdots < j_{|\boldsymbol{y}_1|}$ the indices in U. Since $(\boldsymbol{y}_2)_U = \boldsymbol{y}_1$, for $1 \leq i \leq |\boldsymbol{y}_1|$ it holds that $(\boldsymbol{y}_2)_{\{j_i, \ldots, j_{|\boldsymbol{y}_1|}\}} = (\boldsymbol{y}_1)_{[i\,:\,|\boldsymbol{y}_1|]}$. We will prove, by backward induction that $\{j_1, \ldots, j_{|\boldsymbol{y}_1|}\}$ is a right-most index set. For $i = |\boldsymbol{y}_1|$, if $j_{|\boldsymbol{y}_1|} = |\boldsymbol{y}_2|$ it is clearly a right-most index set. Otherwise, due to the fact that $j_{|\boldsymbol{y}_1|}$ was inserted into U in the third if condition of the algorithm, it must be that $(\boldsymbol{y}_2)_{j_{|\boldsymbol{y}_1|}} \neq (\boldsymbol{y}_2)_{j_{|\boldsymbol{y}_1|}+1}$ (it can not be that $j_{|\boldsymbol{y}_1|} + 1 \in U$ since U is empty). Thus, $j_{|\boldsymbol{y}_1|}$ is the right-most index in its run in $\boldsymbol{y}_2$ and $\{j_{|\boldsymbol{y}_1|}\}$ is a right-most index set. Let us assume that $\{j_{i+1}, \ldots, j_{|\boldsymbol{y}_1|}\}$ is a right-most index. Similarly to the base case, j_i was inserted into U in the third if condition. Hence, it must be that $(\boldsymbol{y}_2)_{j_i} \neq (\boldsymbol{y}_2)_{j_i+1}$ or that $j_i + 1 \in U'$ where U' is the subset of U in the current step of the algorithm. By the induction assumption $\{j_{i+1}, \ldots, j_{|\boldsymbol{y}_1|}\}$ is a right-most index set and if $(\boldsymbol{y}_2)_{j_i} \neq (\boldsymbol{y}_2)_{j_i+1}$, then j_i is the right-most index in its run and $\{j_i, j_{i+1}, \ldots, j_{|\boldsymbol{y}_1|}\}$ is also a right-most index set. Otherwise, j_i and $j_i + 1$ belong to the same run of $\boldsymbol{y}_2$ and $j_i + 1 \in U'$. By the induction assumption all the indices that are greater than $j_i + 1$ and belong to the same run as $j_i + 1$ are also in U', that is, all the indices that are right to j_i and at the same run are in U' and $\{j_i, j_{i+1}, \ldots, j_{|\boldsymbol{y}_1|}\}$ is a right-most index set by definition. We have proven that each index set $U \in \mathcal{U}_{right}$ is a right-most index set, that is, we have proven that $\mathcal{U}_{right} \subseteq \mathcal{U}_{right}(\boldsymbol{y}_1, \boldsymbol{y}_2)$.

To see that $\mathcal{U}_{right}(\boldsymbol{y}_1, \boldsymbol{y}_2) \subseteq \mathcal{U}_{right}$ consider some index set $U \in \mathcal{U}_{right}(\boldsymbol{y}_1, \boldsymbol{y}_2)$ and let $j_1 < \cdots < j_{|\boldsymbol{y}_1|}$ be the indices in U. Since $U \in \mathcal{U}_{right}(\boldsymbol{y}_1, \boldsymbol{y}_2)$, for each i and $j \geq j_i$ we have that $\mathsf{LCS}(i, j) = i$, that is, the fourth condition holds. Consider the path that invokes the first recursive call if and only if $j = j_i$ for some index $1 \leq i \leq |\boldsymbol{y}_1|$, and invokes the second recursive call otherwise. Since U is a right-most index set, one can easily verify that this is a valid path of the algorithm that yields the index set U and append it to the algorithm output U_{right}. ∎

n	The Naive Algorithm	Algorithm 1
50	1558	6.37
75	10449	6.79
100	44652	8.38
125	> 88000	9.46

TABLE I: A comparison of the run time (in seconds) of Algorithm 1 and the naive algorithm to compute $\mathsf{ID}(\boldsymbol{y}_1, \boldsymbol{y}_2, n)$.

We used simulations to evaluate the performance of Algorithm 1. Our simulations worked on 4 different values of $n = \{50, 75, 100, 125\}$. For each n, we created $5,000$ test cases as follows. First, the values of t_1 and t_2 were generated according to the standard normal distribution with mean $\mu = 4$ and standard deviation of $\sigma = 0.5$. Next, $\boldsymbol{y}_2$ was selected randomly from all the sequences of length $n + t_2$. Then, $t_2 + t_1$ bits were selected randomly and deleted from $\boldsymbol{y}_2$ to create $\boldsymbol{y}_1$. We then performed Algorithm 1 and the naive algorithm (described in the begining of this section) to compute $\mathsf{ID}(\boldsymbol{y}_1, \boldsymbol{y}_2, n)$ and evaluated their run time. Both algorithms were implemented in $c{+}{+}$ and performed on our server with Intel(R) Xeon(R) CPU E5-2630 v3 2.40GHz. In all of the tests Algorithm 1 performed the computation of $\mathsf{ID}(\boldsymbol{y}_1, \boldsymbol{y}_2, n)$ significantly faster and improve the speed of the naive algorithm by a factor of more than $5,000$. The results of the tests are summarized in Table I. Note that we performed the comparison over relatively small values of the parameters due to the limitations of the naive algorithm.

REFERENCES

[1] M. Abu-Sini and E. Yaakobi, "On Levenshtein's reconstruction problem under insertions, deletions, and substitutions," under revision to *IEEE Transactions on Information Theory*.

[2] D. Bar-Lev, T. Etzion, and E. Yaakobi, "On Levenshtein balls with radius one," to appear *IEEE International Symposium on Information Theory*, Melbourne, Australia, Jul. 2021.

[3] C. Elzinga, S. Rahmann, and H. Wang. Algorithms for subsequence combinatorics. *Theoretical Computer Science*, 409(3):394–404, 2008.

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

[5] T. Hayashi and K. Yasunaga, "On the list decodability of insertions and deletions," *IEEE International Symposium on Information Theory*, pp. 86–90, Vail, CO, Jun. 2018.

[6] D. S. Hirschberg, "Bounds on the number of string subsequences," *Annual Symposium on Combinatorial Pattern Matching*, pp. 115–122, 1999.

[7] S. Y. Itoga, "The string merging problem," *BIT Numerical Mathematics*, vol. 21, no. 1, pp. 20–30, 1981.

[8] T. Jiang and A. Vardy, "Asymptotic improvement of the Gilbert-Varshamov bound on the size of binary codes," *IEEE Transactions on Information Theory*, vol. 50, no. 8, pp. 1655–1664, Aug. 2004.

[9] V. Junnila, T. Laihonen, and T. Lehtilä, "The Levenshtein's channel and the list size in information retrieval," *IEEE International Symposium on Information Theory*, pp. 295–299, Paris, France, Jul. 2019.

[10] V. I. Levenshtein, "Binary codes capable of correcting deletions, insertions, and reversals," in Doklady Akademii Nauk, vol. 163, no. 4. Russian Academy of Sciences, 1965, pp. 845–848.

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

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

[13] S. Liu, I. Tjuawinata, and C. Xing, "On list decoding of insertion and deletion errors," https://arxiv.org/abs/1906.09705, 2019.

[14] O. Sabary, E. Yaakobi, and A. Yucovich, "The error probability of maximum-likelihood decoding over two deletion/insertion channels," *IEEE International Symposium on Information Theory*, pp. 763–768, Los Angeles, CA, USA, June. 2020.

[15] F. Sala and L. Dolecek, "Counting sequences obtained from the synchronization channel," *IEEE International Symposium on Information Theory*, pp. 2925–2929, Istanbul, Turkey, Jul. 2013.

[16] F. Sala, R. Gabrys, and L. Dolecek, "Gilbert-Varshamov-like lower bounds for deletion-correcting codes," *IEEE Information Theory Workshop*, Hobart, TAS, pp. 147–151, 2014.

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

[18] A. Wachter-Zeh, "List decoding of insertions and deletions," *IEEE Transactions on Information Theory*, vol. 64, no. 9, pp. 6297–6304, 2017.

[19] E. Yaakobi and J. Bruck, "On the uncertainty of information retrieval in associative memories," *IEEE Transactions on Information Theory*, vol. 65, no. 4, pp. 2155–2165, Apr. 2018.

[20] Y. Yehezkeally and M. Schwartz, "Uncertainty of reconstructing multiple messages from uniform-tandem-duplication noise," *IEEE International Symposium on Information Theory*, pp. 126–131, Los Angeles, CA, USA, June. 2020.