Efficient Communication Protocols for Deciding Edit Distance [*]

Hossein Jowhari

MADALGO, University of Aarhus
hjowhari@madalgo.au.dk

Abstract

In this paper we present two communication protocols on computing edit distance. In our first result, we give a one-way protocol for the following Document Exchange problem. Namely given $x \in \Sigma^n$ to Alice and $y \in \Sigma^n$ to Bob and integer k to both, Alice sends a message to Bob so that he learns x or truthfully reports that the edit distance between x and y is greater than k. For this problem, we give a randomized protocol in which Alice transmits at most $\tilde{O}(k \log^2 n)$ bits and each party's time complexity is $\tilde{O}(n \log n + k^2 \log^2 n)$.

Our second result is a simultaneous protocol for edit distance over permutations. Here Alice and Bob both send a message to a third party (the referee) who does not have access to the input strings. Given the messages, the referee decides if the edit distance between x and y is at most k or not. For this problem we give a protocol in which Alice and Bob run a $O(n \log n)$-time algorithm and they transmit at most $\tilde{O}(k \log^2 n)$ bits. The running time of the referee is bounded by $\tilde{O}(k^2 \log^2 n)$. To our knowledge, this result is the first upper bound for this problem.

Our results are obtained through mapping strings to the Hamming cube. For this, we use the Locally Consistent Parsing method of [5, 6] in combination with the Karp-Rabin fingerprints. In addition to yielding non-trivial bounds for the edit distance problem, this paper suggest a new conceptual framework and raises new questions regarding the embeddability of edit distance into the Hamming cube which might be of independent interest.

1 Introduction

For integers m and n, let $ed(x, y)$ denote the standard *edit distance* between two strings x and y from the alphabet $[m] = \{1, \ldots, m\}$ where it is defined as the minimum number of substitutions, insertions and deletions of characters that is required to convert x to y. In this paper, we also consider two variants of this metric. The *Ulam* metric is a submetric of edit distance restricted to sequences with no character repetitions. The *edit distance with moves* [5, 6], denoted by $ed_M(x, y)$, is defined similar to $ed(x, y)$ with addition of a block move operation. Namely moving the entire substring $x[i, j]$ to any location is considered a single operation. To simplify the presentation, throughout this

[*]Part of this research was done while the author was a visiting student at MIT.

paper we assume the alphabet size m is $O(n^c)$ constant c, otherwise we can always use random hashing to make this happen.

Edit distance has been studied in various computational and mathematical models. In this paper we focus on computing edit distance in the context of two communication models. In the first problem, here denoted by DE_k and known as the *Document Exchange* problem [6], there are two communicating parties Alice and Bob respectively holding input strings x and y. We are interested in a one-way communication protocol where Alice sends a message to Bob and Bob either learns x or truthfully reports that $ed(x, y) > k$. Here in this paper, in addition to optimizing the total number of transmitted bits, we are interested in protocols where Alice and Bob both run poly(k, n)-time algorithms.

Protocols for the document exchange problem are of considerable practical importance in communicating data over noisy channels. Consider the following recurring scenario where A transmits a large file x to B over an unreliable link with possible errors of insertion, deletion and substitution of bits. Roughly speaking, using a protocol for the above problem, A can supplement his message with $A(x)$ and avoid retransmissions in the case of small amount of corruption. In a similar situation, these protocols are useful for minimizing communication in the consolidation of distributed data sites. For instance, instead of transmitting their repository, coordinated servers can instead exchange their differences through a document exchange protocol.

For the DE_k problem, we give a communication protocol that transmits $\tilde{O}(k \log^2 n)$ bits while Alice and Bob's running time is bounded by $\tilde{O}(n \log n + k^2 \log^2 n)$. To our knowledge this is the first time-efficient 1-way protocol for this problem [1].

In the second problem, denoted by $ED^{\uparrow}_{k,k+1}$, we are interested in *sketching* protocols for deciding edit distance. Namely, we would like to have an efficiently computable mapping $h_k : X \to \{0,1\}^t$ with $t = \text{poly}(k, \log n)$, such that having access to only $h_k(x)$ and $h_k(y)$, there is an efficient algorithm that decides if $d(x, y) \leq k$ or not. Equivalently, defined in terms of the *simultaneous* protocols [11], Alice holds x, and Bob holds y but they are only allowed to send one message to a third party named the Referee whose duty is to decide $ed(x, y) \leq k$ without any access to the input strings. Likewise here we are interested in protocols that run time-efficient computations. In this problem (and in DE_k as well), we assume the communicating parties have access to a shared source of randomness.

The sketching protocols in addition to their application in data communication, have implications in classification and data structures for Nearest Neighbour search. In particular consider a scenario where we would like to find two strings with edit distance bounded by k in a collection of m strings. Using a sketching protocol for the above problem, one can first build sketches of the strings and then compare the short sketches instead of running an edit distance algorithm over long strings.

For the problem of $ED^{\uparrow}_{k,k+1}$ restricted to non-repeating strings, we give a protocol that transmits at most $\tilde{O}(k \log^2 n)$ bits. The running time of all communicating parties is bounded by $\tilde{O}(k^2 \log^2 n)$. Unfortunately our method does not yield an upper bound for standard edit distance. It remains an open problem whether there exists a

[1] In fact a time-efficient 1-way protocol that transmits $O(k \log k \log(n/k))$ bits can inferred from the work of Irmak et al. [9] which uses a different method. The author was not aware of this result in time of the submission of this paper.

simultaneous protocol for deciding edit distance over general strings.

Previous and Related Works. Also known as the *Remote File Synchronization* [9], the Document Exchange problem DE_k has been studied extensively by Cormode *et al.* [6]. Conditioning on $ed(x,y) \leq k$, it has been shown there is a deterministic one-way protocol for DE_k, resulting from the graph coloring method of [14], that transmits only $O(k \log n)$ bits. However in this protocol, Bob's running time is exponential in k. Similarly in a randomized protocol for this problem, Alice builds a random hash $h(x)$ and send it to Bob. Bob, using the same source of randomness, compares $h(x)$ with the hash value of all the strings within distance k from y. If he finds a match, x has been learned otherwise he can declare that $ed(x,y) > k$. It can be shown that random hashes of $O(k \log n)$ bit length are enough so that Bob succeeds with high probability. Once again Bob's running time will be exponential in k as he should try $n^{O(k)}$ number of strings.

However if Alice and Bob are allowed to use a multi-round protocol there is a time-efficient protocol. In [6] Cormode *et al* have given a time-efficient solution that transmits $O(k \log(n/k) \log k)$ bits in total and runs in $O(\log n)$ rounds. This protocols works through recursively dividing the string into $O(k)$ blocks and detecting the differences through comparing the fingerprints of each block. Building on similar ideas, Irmak *et al.* [9] have a shown a single round $O(k \log(n/k) \log n)$ protocol for the document exchange problem under edit distance with moves.

To our knowledge, prior to our work, there was no non-trivial upper bound on the sketching complexity of deciding edit distance. This was even true assuming computationally unbounded communicating parties. On the lower bound side, it is known that any protocol for DE_k needs to transmit $\Omega(k)$ bits. This follows from a lower bound for Hamming distance (see the concluding discussions in [3]) This fact remains true when Alice and Bob's inputs are non-repeating strings [8, 17]. Needless to say, this lower bound also applies to the sketching variant since Bob can take up the role of the referee.

Considering the L_p metric and in particular the Hamming metric, both the document exchange and the sketching problem are fairly well-understood. In fact solutions for both of these problems can be derived from any exact sparse recovery scheme (see [7] for an introduction to sparse recovery). Here the L_p differences between x and y can be inferred from the measurements Ax and Ay. Considering that there are sparse recovery matrices with $O(k \log(n/k))$ number of rows, this leads to an $O(k \log(n/k) \log n)$ bits bound for our problems. There are however non-linear methods that give better bounds. In particular for Hamming distance we use the following result in our work.

Lemma 1 *[16] Let x and y be two points in $[m]^n$. There exists a randomized mapping $s_k : [m]^n \rightarrow \{0,1\}^{O(k \log n)}$ such that given $s_k(x)$ and $s_k(y)$, there is an algorithm that outputs all the triples $\{(x_i, y_i)\}$ satisfying $x_i \neq y_i$ in $O(k \log n)$ time. For l-sparse vectors x, the sketch $s_k(x)$ can be constructed in $O(l \log n)$ time and space in one pass.*

The best exact algorithm for edit distance runs in $O(n^2/\log^2 n)$ time [13]. For the decision version of the problem, there is an algorithm by [12] that runs in $O(n + k^2)$ time. In contrast with little progress over the exact complexity, there is a long sequence of results on approximating edit distance. The best near-linear time approximation algorithm for edit distance [2] runs in $O(n^{1+\epsilon})$ time and outputs an $O(\log^{O(1/\epsilon)} n)$ approximation. The sketching complexity of the gap questions regarding edit distance has been studied directly or implicitly in [4, 3, 1]. In particular [3] gives a $O(1)$-size sketch for distinguishing between k versus $(kn)^{2/3}$. We refer the reader to [2] for a detailed history on these results.

Techniques and ideas. In this paper we propose the following general framework that is applicable to all distance functions. Given $d : [m]^n \times [m]^n \to \mathbb{R}^+$, we design a mapping f from distance d to Hamming. Formally let $f : [m]^n \to \{0,1\}^{n'}$ having the following properties.

1. n' is bounded by $n^{O(1)}$ and for all x, $f(x)$ can be computed efficiently.

2. There exists $1 \leq e_f << n$ such that for distinct u and v, we have $0 < \mathcal{H}(f(u), f(v)) \leq e_f d(u,v)$ with probability at least 7/8.

3. There is a polynomial time algorithm R_f where given $f(x)$ obtains x exactly with probability at least 7/8.

Using the mapping f and the reconstruction procedure R_f, we propose the following protocol. In the beginning, Alice and Bob both compute $f(x)$ and $f(y)$, and then they run a document exchange protocol under Hamming distance over $f(x)$ and $f(y)$ with parameter $k' = O(e_f k)$ (see Lemma 1). It follows that if $d(x,y) \leq k$, Bob will learn $f(x)$ and subsequently using R_f he reconstructs x. Considering the bounds from Lemma 1, communication complexity of this solution is bounded by $O(e_f \cdot k \cdot \log n)$ while the running time depends on the construction time of $f(x)$ and the time complexity of R_f.

We remark that, we do not need to use a mapping with low contraction as apposed to the standard metric embeddings. As long as distinct strings are mapped to different vectors and they are efficiently retrievable, the contraction of distances is not important. Naturally deriving such mappings is considerably easier in comparison with the standard low distortion embeddings. In fact, as we see in Section 2, we almost directly obtain a mapping from the Cormode-Muthukrishnan's embedding of *edit distance with moves* into L_1 [5]. This works because edit distance with moves is upper bounded by edit distance and moreover the expansion factor of its mapping is $O(\log n \log^* n)$ which is considerably better than the existing bounds for edit distance [2]. The CM's embedding is quite efficient. It constructs a parsing tree over the input in near-linear time and encodes the substrings that are marked in the process into $f(x)$. As we shall see, to complete the picture we just need to equip our mapping with an efficient reconstruction procedure.

Roughly speaking, our reconstruction algorithm, having collected the Rabin-Karp fingerprints of the substrings of x that were obtained from the encoding $f(x)$, it rebuilds the original parsing tree that was used to create $f(x)$. As we shall see, this can be done in a fairly straightforward manner. Once the parsing tree is reconstructed the original string x can be inferred from the labels of the leaves of this tree.

[2]The best low distortion embedding for edit distance, due to Ostrovsky and Rabani [15], has $2^{\Omega(\sqrt{\log n})}$ as both the expansion and contraction factor.

In our sketching result for deciding Ulam, similar to the solution of the document exchange, Alice and Bob use the mapping f to create $f(x)$ and $f(y)$. Granted $ed(x,y)$ is bounded, the referee will learn $f(x)-f(y)$ and from this difference he will be able to construct partial parsing trees corresponding to x and y. It will be shown that the partial trees are enough to decide if $ed(x,y) \leq k$ or not. This last step of deciding the edit distance from partial recovery of x and y relies on the non-repetitiveness of the input sequences.

2 A 1-way protocol for Document Exchange

Before presenting our solution for edit distance, as a warm-up and along the lines we described above, we present a simple solution for the *Ulam metric*. The mapping f_U for Ulam distance is defined as follows. Let π be a non-repeating string. Corresponding to each pair (i,j) where $i,j \in \Sigma$, $f_U(\pi)$ has a coordinate. We set $f_U(\pi)_{i,j} = 1$ if j appears right after i in π otherwise we set $f_U(\pi)_{i,j} = 0$. It can be verified easily that the expansion factor of f_U is bounded by a constant ($e_{f_U} \leq 8$). Moreover the reconstruction procedure R_{f_U} is straightforward. This gives us the following lemma.

Lemma 2 *There is a one-way protocol for DE_k restricted to permutations that transmits $O(k \log n)$ bits and its running time is bounded by $O(n \log n)$.*

To define our mapping for general strings, we use the following result by Cormode and Muthukrishnan [5].

Lemma 3 *There is a mapping $g : \Sigma^n \to [n]^l$, such that for every pair $u,v \in \Sigma^n$, we have*

$$\frac{1}{2} ed_M(u,v) \leq \| g(u) - g(v) \|_1 \leq O(\log n \log^* n) ed_M(u,v).$$

Moreover the number of non-zero entries in $g(u)$ is bounded by $2n$.

Unfortunately the mapping g generates exponentially long vectors, i.e $l = 2^{O(n \log m)}$, and thus it is unsuitable for a direct application. However since the generated vectors are very sparse, we deploy random hashing to reduce its dimensionality. In the following we proceed with a brief background on g and then we describe the random hash function (supplemented with some extra information) that gives us the desired mapping f. Finally we present the reconstruction algorithm R_f.

The ESP tree. The embedding of [5] uses a procedure called *Edit Sensitive Parsing* (ESP) to build a rooted tree $T_x = (V_x, E)$ over the input string x where the leaves of T_x are the single characters of x. Each node $v \in V_x$ *represents* a unique substring $x[i,j]$. This means the characters $x(i)x(i+1)\ldots x(j)$ are at leaves of the subtree rooted at v. See Figure 3 for a pictorial representation. We let $s(v)$ denote the substring that v represents. Every non-leaf node of T_x has either 2 or 3 children. We introduce the following notations. We denote the left-hand child of v by v_l, the middle child (if exists) by v_m and the right-hand child by v_r.

We need the following fact regarding the properties of the ESP tree; its complete proof can be inferred from the details in [5].

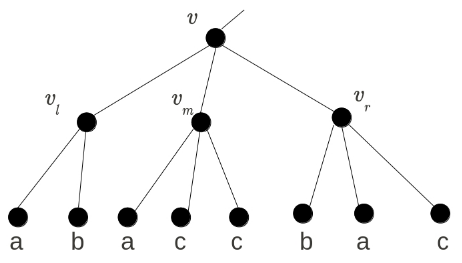

Figure 1: A pictorial presentation of a part of an ESP tree.

Lemma 4 *The number of structurally distinct subtrees in T_x that represent the same (but arbitrary) $\alpha \in \Sigma^*$ is bounded by $2^{O(\log^* n)}$.*

PROOF: Let $v \in V_x$ and let $x[i, j]$ be the substring represented by v. From the procedure ESP, we have that the tree rooted at v depends only on the content of a substring $x[i', j']$ where $i' = i - O(1)$ and $j' = j + O(\log^* n)$. $\square$

Given T_x, the vector $g(x)$ is defined as follows. Corresponding to each string α of length n, $g(x)$ has a coordinate. Let $g(x)[\alpha]$ represent the coordinate corresponding to α. We set $g(x)[\alpha]$ to be the number of nodes in T_x that represent α. This finishes the description of g.

Definition of f. We use a linear hash function to map the coordinates of g to polynomially bounded numbers. For the choice of our hash function we use the classical Rabin-Karp fingerprinting method [10]. Let $q > 4n^4$ be a prime and $r \in \mathbb{Z}_q^*$ be randomly selected from $[q]$. For $\alpha \in \Sigma^*$, the Rabin-Karp fingerprint of α is defined as $\Phi(\alpha) = \sum_{i=1}^{|\alpha|} \alpha[i] \cdot r^{i-1} \pmod q$. The following facts are well-known and the reader is referred to [10] for the proofs.

(F1) $\Phi(\alpha)$ can be computed in one pass over α using $O(\log n)$ bits of space.

(F2) Let $\beta \neq \alpha$ be a string of length at most n. $\Pr_r[\Phi(\alpha) = \Phi(\beta)] \leq \frac{1}{n^3}$.

(F3) We have $\Phi(\alpha \circ \beta) = \Phi(\alpha) + r^{|\alpha|+1}\, \Phi(\beta) \pmod q$ where the arithmetic operations are done in $\mathbb{Z}_q$.

Note 5 *Since with high probability there will be no collision between the fingerprint of different strings, to simplify the presentation, in the rest of this paper we will assume that is the case.*

Now we are ready to introduce $f : \Sigma^n \to [n] \times [q]^4$. Given $x \in \Sigma^n$, we set the coordinates of $f(x)$ as follows. For a non-leaf $v \in T_x$ with three children v_l, v_m and v_r, we set $f(x)[|s(v)|, \Phi(s(v)), \Phi(s(v_l)), \Phi(s(v_m)), \Phi(s(v_r))]$ to be the number of subtrees in T_x that represent $s(v)$ and their root substring is partitioned into three blocks of $s(v_l)$, $s(v_m)$ and $s(v_r)$. We do the same for the nodes with two children except that we regard the middle block $s(v_m)$ as the empty string with $\Phi(s(v_m)) = 0$. For the leaf $v \in T_x$, we set $f(x)[1, \Phi(s(v)), 0, 0, 0]$ to be the number of the occurrences of the character $s(v)$ in x. The rest of the coordinates in $f(x)$ are set to zero.

The Reconstruction Algorithm
Input: $f(x)$, **Output:** x

Initialization: Let S_x denote the set of non-zero coordinates in $f(x)$, *i.e*

$$S_x = \{\ (l, j_0, j_1, j_2, j_3) \mid f(x)[l, j_0, j_1, j_2, j_3] \neq 0\ \}.$$

Begin with an empty tree $T'_x = (V', E')$. Find $v \in S_x$ with $l = n$ and add it to V'. Let $S_x = S_x / \{v\}$. Note that such a node is unique since there is only one string with length n. This is going to be the root of T'_x. In the following, abusing the notation, we let $\Phi(v)$ denote the second coordinate of v.

Iteration: Repeat until S_x is empty.

1. Pick an arbitrary leaf $v \in T'_x$ which does not correspond to a single character.
2. Given the non-zero fingerprints j_1, j_2 and j_3 corresponding to the children of v, find the elements in v_l, v_m and v_r in S_x where $\Phi(v_l) = j_1$, $\Phi(v_m) = j_2$ and $\Phi(v_r) = j_3$. Break the ties arbitrarily.
3. Let $S_x = S_x / \{v_l, v_m, v_r\}$ and add the excluded elements to T'_x as the children of v. Note that if $j_2 = 0$, we would only add v_l and v_r.

Finalization: Counting from left to right, let x_i be the character that is represented by the i-th leaf of T'_x.

Figure 2: Description of the reconstruction algorithm R_f.

Proposition 6 *Let $u, v \in \Sigma^n$. We have $\mathcal{H}(f(u), f(v)) \leq 2^{O(\log^* n)} \parallel g(u) - g(v) \parallel_1$.*

PROOF: Let S_u and S_v be the set of substrings that are represented by the nodes in the trees T_u and T_v respectively. Assuming the mapping Φ produces distinct values over $S_u \cup S_v$ and by Lemma 4, it follows there are at most $2^{O(\log^* n)}$ non-zero coordinates in f corresponding to a substring α. This completes the proof of the lemma. $\Box$

The above proposition and Lemma 3, give us the following bound on the expansion factor of f.

Corollary 7 $e_f = O(\log n \cdot 2^{O(\log^* n)} \cdot \log^* n)$.

Reconstruction algorithm R_f. The description of this algorithm is given in Figure 2. Our procedure R_f works by constructing a tree T'_x that has the same sequence of (labeled) leaves as T_x does and as result we reconstruct x. The tree is constructed in a top-down fashion starting from the root of T_x and expanding the leaves by finding the encodings of its children in the vector $f(x)$. We point out that T'_x might not be structurally equivalent with T_x but it is guaranteed that the leaves are in the same labeling order.

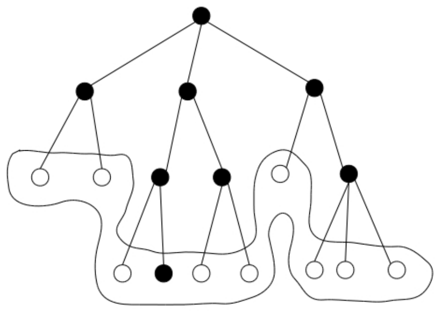

Figure 3: A pictorial presentation of the tree $T(\mathcal{V}_X)$ shown in black nodes. The curved area shows the nodes in Γ_X.

Theorem 8 *There is a randomized one-way protocol for DE_k that transmits $\tilde{O}(k \log^2 n)$ bits and succeeds with high probability. The running time of each party is bounded by $O(n \log n + k^2 \log^2 n)$.*

Proof: (Sketch) The correctness of the result follows from Lemma 1, the general framework that we introduced in the introduction and the description of the the reconstruction procedure. The running time of creating $f(x)$ is bounded by $O(n \log n)$ since the vector has at most $O(n)$ non-zero coordinates. The reconstruction procedure performed in a naive manner takes $O(n^2)$ time as finding each child of node might take linear time. To expedite this procedure, we do not build T'_x from the scratch. Since the trees T_x and T_y differ on at most $\tilde{O}(k \log n)$ nodes, we can start from T_y (after pruning its disagreements with T_x) and as result we need to process at most $\tilde{O}(k \log n)$ entries. We defer details of this improvement to the long version of this paper.

Having obtained x, Bob can decide $ed(x, y) \le k$ in $O(n + k^2)$ time using the algorithm from [12]. This finishes the proof. □

3 A sketching result for Ulam

Let S_n be the set of permutations over $\{1, \dots, n\}$. In this section we present our sketching result for edit distance over S_n. The following lemma is the heart of this result.

Lemma 9 *Let $x, y \in S_n$. Let f be the mapping described as before. Given the non-zero coordinates of $\Delta_{x,y} = f(x) - f(y)$, there is a $\tilde{O}(k^2 \log^2 n)$ algorithm that decides whether $ed(x, y) \le k$ or not.*

Proof: First we observe that since x has no repetitions, $f(x)$ is a zero-one vector. Let

$$X = \{\ i\ |\Delta_{x,y}(i) = 1\}.$$

The set X points to the substrings of T_x that are not represented by T_y. Similarly we define the set Y where it corresponds to the negative coordinates in $\Delta_{x,y}$.

Let $\mathcal{V}_X$ be the set of nodes in T_x that corresponds to X. Clearly $root(T_x) \in \mathcal{V}_X$. Now let $T(\mathcal{V}_X)$ be the rooted tree that is obtained by attempting to reconstruct T_x starting from $root(T_x)$. To build $T(\mathcal{V}_X)$, starting from the root, we try to expand every node until it reaches down to a single character (*i.e.* to a leaf of T_x) or its children are entirely missing in $\mathcal{V}_X$ because they have identical counterparts in T_y. Note that it is possible that one or two children are missing in $\mathcal{V}_X$. In that case, since we have supplemented every node with the fingerprint of its children, a node can still be expanded. It is also possible that some nodes in $\mathcal{V}_X$ are left unpicked after the end of this process. Those nodes represent substrings whose parents exist in both T_x and T_y but they are partitioned differently and hence have been mapped to different coordinates in $f(x)$ and $f(y)$. These nodes will be neglected.

Let Γ_X represent the set of vertices in $T_x/\mathcal{V}_X$ that are the immediate neighbors of the nodes in $T(\mathcal{V}_X)$. We also include the leaves of $T(\mathcal{V}_X)$ which represent single characters into Γ_X. Note that by the non-repetitiveness of x, $T(\mathcal{V}_X)$ is indeed a partial subtree of T_x rooted at $root(T_x)$ and hence Γ_X is well-defined. Also since for each node we have stored the information regarding its children, the set Γ_X can be computed. It should be clear that Γ_X gives a non-overlapping partitioning of x into $\tilde{O}(k \log n)$ blocks. By the definition, every block in Γ_X is identical to a represented substring in T_y. We perform the same process for string y and obtain Γ_Y.

We cannot claim that there exists a perfect matching between Γ_X and Γ_Y, however we can find a perfect matching between sets of consecutive blocks. To see this, let C be the longest block in $\Gamma_X \cup \Gamma_Y$ (breaking the ties arbitrarily) and w.l.o.g assume $C \in \Gamma_X$. Since C has an identical match in T_y it must match a set of consecutive blocks in Γ_Y. We pick this matching and continue with finding a matching for the longest block in the remained unmatched ones. It follows from the definition of Γ_X and Γ_Y and fact that there are no repeating substrings, every block will be matched at the end. Moreover such a mapping can be found in $O(k^2 \log^2 n)$ time.

Naturally, the mapping between the consecutive blocks in Γ_X and Γ_Y defines a perfect matching between the indices in x and y. Let $h(i)$ denote the index where x_i is mapped to. We create two permutations x' and y' so that x'_i and $y'_{h(i)}$ receive the same label. Clearly $ed(x', y') = ed(x, y)$ since relabeling does not affect the edit distance. It follows that we can compute the edit distance between x and y. This last step performed in a naive manner takes $O(n \log n)$ time but considering the fact that in an optimal alignment between x' and y' a block is either entirely aligned or it is deleted, we can find an optimal alignment using a dynamic programming for a weighted encoding of the problem. Therefore this can also be performed in $\tilde{O}(k^2 \log^2 n)$ time. □

The above lemma combined with Lemma 1 give us the following result.

Theorem 10 *There is a randomized mapping $u_k : S_n \to \{0,1\}^{O(k \log^2 n)}$ such that given $u_k(x)$ and $u_k(y)$ for $x, y \in S_n$, there is an $\tilde{O}(k^2 \log^2 n)$-time algorithm that decides whether $ed(x, y) \leq k$ or not.*

4 Concluding remarks

1. In our protocol for the Document Exchange problem, we used a randomized mapping $f : (\text{ed}, [m]^n) \to (\text{Hamming}, \{0,1\}^{n'})$ with polynomially large n' and $e_f = \tilde{O}(\log n)$. Is there a similar mapping with $e_f = o(\log n)$? Such a mapping equipped with a polynomial time reconstruction procedure results in an improved protocol for the DE_k problem. On the other hand, given that such a mapping exists for the Ulam metric (the mapping f_U in Section 2), showing the impossibility of a similar fact for the edit distance will result in proving an interesting seperation theorem between the two metrics. From the angel of the low distortion embeddings, seperating Ulam from edit distance over repetitive strings has yet remained an open question.

2. In our sketching result for the Ulam metric, we have not used the simpler mapping f_U of Section 2. This is because it does not preserve the edit distance. In other words, there are pairs of strings (x_1, y_1) and (x_2, y_2) such that $f_U(x_1) - f_U(y_1)$ and $f_U(x_2) - f_U(y_2)$ are identical while $ed(x_1, y_1)$ and $ed(x_2, y_2)$ are arbitrarily far apart.

3. The sketching result for Ulam can be generalized to the case when only one of the strings is a permutation. This is true since we can still relabel the content of each mapped block with arbitrary characters. Also we may not have a perfect matching but the content of the blocks that aren't mapped will be revealed. In the case of general strings, we can also obtain a mapping between the blocks of the input strings. However, because of repetitions, it is not clear how we can use this information to learn the edit distance.

4. The sketching algorithm of Section 3 can be adapted to work as a streaming algorithm over interleaving input sequences. This follows from the fact that Lemma 1 is a streaming result. Moreover since ESP tree can be built in a streaming fashion (see Section 4.3 in [5]), we are able to derive a streaming algorithm.

Acknowledgement. The author would like to thank Ronitt Rubinfeld for kindly hosting his visit to MIT. Thanks also to Cenk Sahinalp, Mert Sağlam, Qin Zhang and Djamal Belazzougui for useful discussions. In particular thanks to Djamal Belazzougui for bringing [9] to the author's attention.

References

[1] Alexandr Andoni and Robert Krauthgamer. The computational hardness of estimating edit distance [extended abstract]. In *FOCS*, pages 724–734, 2007.

[2] Alexandr Andoni, Robert Krauthgamer, and Krzysztof Onak. Polylogarithmic approximation for edit distance and the asymmetric query complexity. In *FOCS*, pages 377–386, 2010.

[3] Ziv Bar-Yossef, T. S. Jayram, Robert Krauthgamer, and Ravi Kumar. Approximating edit distance efficiently. In *FOCS*, pages 550–559, 2004.

[4] Tugkan Batu, Funda Ergün, Joe Kilian, Avner Magen, Sofya Raskhodnikova, Ronitt Rubinfeld, and Rahul Sami. A sublinear algorithm for weakly approximating edit distance. In *STOC*, pages 316–324, 2003.

[5] Graham Cormode and S. Muthukrishnan. The string edit distance matching problem with moves. *ACM Transactions on Algorithms*, 3(1), 2007.

[6] Graham Cormode, Mike Paterson, Süleyman Cenk Sahinalp, and Uzi Vishkin. Communication complexity of document exchange. In *SODA*, pages 197–206, 2000.

[7] Anna Gilbert and Piotr Indyk. Sparse recovery using sparse matrices. In *Proceedings of IEEE*, 2010.

[8] Parikshit Gopalan, T. S. Jayram, Robert Krauthgamer, and Ravi Kumar. Estimating the sortedness of a data stream. In *SODA*, pages 318–327, 2007.

[9] Utku Irmak, Svilen Mihaylov, and Torsten Suel. Improved single-round protocols for remote file synchronization. In *INFOCOM*, pages 1665–1676, 2005.

[10] Richard M. Karp and Michael O. Rabin. Efficient randomized pattern-matching algorithms. *IBM Journal of Research and Development*, 31(2):249–260, 1987.

[11] Eyal Kushilevitz and Noam Nisan. *Communication complexity.* Cambridge University Press, 1997.

[12] Gad M. Landau, Eugene W. Myers, and Jeanette P. Schmidt. Incremental string comparison. *SIAM J. Comput.*, 27(2):557–582, 1998.

[13] William J. Masek and Mike Paterson. A faster algorithm computing string edit distances. *J. Comput. Syst. Sci.*, 20(1):18–31, 1980.

[14] Alon Orlitsky. Interactive communication: Balanced distributions, correlated files, and average-case complexity. In *FOCS*, pages 228–238, 1991.

[15] Rafail Ostrovsky and Yuval Rabani. Low distortion embeddings for edit distance. *J. ACM*, 54(5), 2007.

[16] Ely Porat and Ohad Lipsky. Improved sketching of hamming distance with error correcting. In *CPM*, pages 173–182, 2007.

[17] Xiaoming Sun and David P. Woodruff. The communication and streaming complexity of computing the longest common and increasing subsequences. In *SODA*, pages 336–345, 2007.