Electronic Colloquium on Computational Complexity, Report No. 111 (2015)

Low Distortion Embedding from Edit to Hamming Distance using Coupling

Diptarka Chakraborty* Elazar Goldenberg† Michal Koucký ‡

July 8, 2015

Abstract

The Hamming and the edit metrics are two common notions of measuring distances between pairs of strings x, y lying in the Boolean hypercube. The edit distance between x and y is defined as the minimum number of character insertion, deletion, and bit flips needed for converting x into y. Whereas, the Hamming distance between x and y is the number of bit flips needed for converting x to y.

In this paper we study a randomized injective embedding of the edit distance into the Hamming distance with a small distortion. This question was studied by Jowhari (ESA 2012) and is mainly motivated by two questions in communication complexity: the document exchange problem and deciding edit distance using a sketching protocol.

We show a randomized embedding with quadratic distortion. Namely, for any x, y satisfying that their edit distance equals k, the Hamming distance between the embedding of x and y is $O(k^2)$ with high probability. This improves over the distortion ratio of $O(\log n \log^* n)$ obtained by Jowhari for small values of k. Moreover, the embedding output size is linear in the input size and the embedding can be computed using a single pass over the input.

*Department of Computer Science & Engineering, Indian Institute of Technology Kanpur, Kanpur, India, diptarka@cse.iitk.ac.in. Work done while visiting Charles University in Prague.

† Charles University in Prague, Computer Science Institute of Charles University, Malostranské námesti 25, 118 00 Praha 1, Czech Republic, elazargold@gmail.com. The research leading to these results has received funding from the European Research Council under the European Union's Seventh Framework Programme (FP/2007-2013) / ERC Grant Agreement n. 616787.

‡Charles University in Prague, Computer Science Institute of Charles University, Malostranské námesti 25, 118 00 Praha 1, Czech Republic, koucky@iuuk.mff.cuni.cz. The research leading to these results has received funding from the European Research Council under the European Union's Seventh Framework Programme (FP/2007-2013) / ERC Grant Agreement n. 616787.

ISSN 1433-8092

1 Introduction

The *Hamming* and the *edit distance* (aka *Levenshtein distance*) [Lev66] are two common ways of measuring distances between pairs of strings $x, y \in \{0,1\}^n$. The edit distance between x and y, denoted by $\Delta_e(x,y)$, is defined as the minimum number of character insertion, deletion, and bit flips needed for converting x into y and the Hamming distance between x and y, denoted by $\Delta_H(x,y)$, is the number of bit flips needed for converting x to y.

In this paper we study the question of *randomized embedding* from edit distance metric into Hamming metric with a small distortion. Given two metric spaces (X_1, d_1) and (X_2, d_2), an embedding from X_1 into X_2 with *distortion factor* ϕ_d is a mapping $\phi : X_1 \to X_2$, such that $\forall_{u,v \in X_1}$, $d_2(\phi(u), \phi(v)) \leq \phi_d \cdot d_1(u,v)$. In general, it is always interesting to find an embedding with small distortion from a "not so well" understood metric space to a metric space for which efficient algorithms are known, because this directly provides us efficient algorithms for the original metric space. Our main result provides a linear-time embedding with quadratic distortion between input edit distance and output Hamming distance.

Our main motivation behind the study of this question is computing edit distance in the context of two communication models. The first problem, known as *Document Exchange* problem [CPSV00], where two communication parties Alice and Bob hold two input strings x and y and based on the message transmitted by Alice, Bob's task is to decide whether $\Delta_e(x,y) > k$ and if $\Delta_e(x,y) \leq k$ then to report x correctly. We assume that both the parties have access to a shared randomness and also consider an added restriction that both Alice and Bob run in time polynomial in n and k. The document exchange problem with this restriction on running time was also studied in [Jow12] and we will discuss about what is known about this problem in the later part of this section.

Another important problem related to embedding is to decide edit distance using a *sketching* protocol. In this problem, given two strings x and y, we would like to compute *sketches* $s(x)$ and $s(y)$ or in other words a mapping $s : \{0,1\}^n \to \{0,1\}^t$ such that t is much smaller compared to n and our objective is to decide whether $\Delta_e(x,y) > k$ or not having access to only $s(x)$ and $s(y)$. One can also view the same problem as a two-party public-coin *simultaneous* communication protocol [KN97], where Alice holds x and Bob holds y and both of them are only allowed to send one message to a third referee whose job is to decide whether $\Delta_e(x,y) > k$ or not depending on the messages he receives from Alice and Bob.

1.1 Our Main Result

Our main result shows the existence of a randomized mapping f from edit distance into Hamming distance that at most squares the edit distance. Our result applies to the shared randomness model. Namely, we show that for every pair of strings $(x,y) \in \{0,1\}^n$ having a small edit distance the following holds. The Hamming distance between the encoded strings is small, provided that the encoding was done using the same sequence of random bits[1], formally:

Theorem 1.1. *For any integer $n > 0$, there is $\ell = O(\log n)$ and a function $f : \{0,1\}^n \times \{0,1\}^\ell \to \{0,1\}^{3n}$ such that for any pair of strings $x, y \in \{0,1\}^n$:*

$$\frac{1}{2} \cdot \Delta_e(x,y) \leq \Delta_H(f(x,r), f(y,r)) \leq O((\Delta_e(x,y))^2)$$

[1] Our result can easily be extended for strings lying in a larger alphabet (See Section 5).

with probability at least $2/3$ *over the choice of the random string* $r \in \{0,1\}^\ell$*. Furthermore,* f *can be computed in a streaming fashion in (*non-uniform*) log-space using one pass over the input string* x*.*

We can also give a uniform version of this theorem by allowing a little bit more random bits.

Theorem 1.2. *There is an algorithm* A *computing a function* $f : \{0,1\}^n \times \{0,1\}^\ell \to \{0,1\}^{3n}$ *where* $\ell = O(\log^2 n)$ *such that for any pair of strings* $x, y \in \{0,1\}^n$*:*

$$\frac{1}{2} \cdot \Delta_e(x,y) \leq \Delta_H(f(x,r), f(y,r)) \leq O((\Delta_e(x,y))^2)$$

with probability at least $2/3$ *over the choice of the random string* $r \in \{0,1\}^\ell$*. The algorithm* A *on input* x *and* r *works in a streaming fashion in log-space using one pass over its input* x *and random access to* r*. On a word RAM with word size* $O(\log n)$*,* A *can be implemented so that it uses only constant amount of working memory, i.e.,* $O(1)$ *words, and uses* $O(1)$ *amortized time per output bit (*$O(\log n)$ *per output bit in the worst-case.)*

In both of the above theorems, with probability at least $1 - 1/n^{\Omega(1)}$ over the choice of random r, x can be recovered from $f(x,r)$. Indeed, the algorithm computing f knows which is the case by the end of its computation. Hence, the function f in both of the theorems can be made one-to-one by appending r at the end of the output and also appending either 0^n or x depending on whether x can be recovered or not. (One would require an extra pass over x to append it.) Since the failure probability of the recovery is low, x would be appended only with low probability and hence, with high probability this would not affect the Hamming distance between $f(x,r)$ and $f(y,r)$. (Most of the times we append 0^n on both x and y.) This answers positively Problem 59 of Dortmund Workshop on Algorithms for Data Streams 2012 [DW112] in the randomized case.

We can specify the trade-off between the Hamming distance and its probability: for any positive $c \in \mathbb{R}$, the probability that $\Delta_H(f(x,r), f(y,r)) \leq c \cdot \Delta_e(x,y)^2$ is at least $1 - \frac{12}{\sqrt{c}} - O(\frac{1}{n})$ (extra $O(1/n)$ term comes from the error incurred due to Nisan's PRG discussed in Section 4). On the other hand $\Delta_H(f(x,r), f(y,r)) \geq \Delta_e(x,y)/2$ happens with probability $1 - 1/n^{\Omega(1)}$.

One may naturally wonder what is the distribution of the resulting Hamming distance. It very much depends on the two strings x and y. For example if y is obtained from x by flipping the first k bits then with high probability the Hamming distance of $f(x,r)$ and $f(y,r)$ is $O(k)$. On the other hand, if y is obtained from a *random* x by flipping each n/k-th bit (where $n \geq k^3$) then with high probability the Hamming distance of $f(x,r)$ and $f(y,r)$ will be $\Omega(k^2)$. Interestingly though, for any two distinct fixed x and y, the expectation of $\Delta_H(f(x,r), f(y,r))$ is $\Theta(n)$ so the distribution is heavy tailed. For many applications this is not any problem as in many applications one can abort when the Hamming distance is large and retry with a new random seed.

Our embedding protocol is as follows: First we pick (using the random string r) a sequence of random functions $h_1, \ldots, h_{3n} : \{0,1\} \to \{0,1\}$. We further maintain a pointer i for current position on the input string x. In time $t \leq 3n$ we append the bit x_i to the output, and increment i by $h_t(x_i)$ (if i exceeds n, we pad the output string with zeros).

Let us shed some light on the intuition behind the distortion guarantee. It is a consequence of coupling of Markov chains. Let $x, y \in \{0,1\}^n$ and $k = \Delta_e(x,y)$. Consider the strings $f(x,r), f(y,r)$ (i.e. the strings are encoded using the same sequence $h_1, \ldots, h_{3n}$). Observe that as long as the pointer i points to indices in the shared prefix of x and y, the two output strings are equal. In the first iteration when i points to an index such that $x_i \neq y_i$, the output bits become different. Nevertheless, the increment of i is done independently in the embedding of each input. This independent increment

is done in each iteration where the output bits are different. The crux of our argument relies on this independent increment to ensure that with high probability $\Delta_H(f(x,r), f(y,r)) \leq ck^2$. This is done by reducing our problem to a certain question regarding random walks on the integer line.

Algorithm of Saha [Sah14]: An idea similar to ours is used in the algorithm of [Sah14] for computing the edit distance of a string of parenthesis of various types from a well parenthesized expression. The main idea of the algorithm presented in [Sah14] is to process the string left to right, push opening parenthesis on a stack and match them against closing parenthesis. Whenever there is a mismatch remove at random either the closing parenthesis or the opening one. This algorithm can be applied also to approximately compute the edit distance of strings by pushing a reverse of one of the strings on the stack and matching the other string against the stack. Whenever there is a mismatch remove at random a symbol either from the top of the stack or from the other string.

In order to approximate the edit distance of two strings at hand it is fairly natural idea to remove the mismatched symbols at random. Our algorithm also builds on this idea. However, there is a major technical challenge when we do not have access to both of the strings at the same time and we should remove the mismatched characters. We do not know which one are those. Deleting symbols at random is unlikely to provide a good result. Hence, it is not clear that one can carry out this basic idea in the case of string embedding when we have access only to one of the strings. Indeed, one needs some *oblivious synchronization mechanism* that would guarantee that once the removal process is synchronized it stays synchronized. The existence of such a mechanism is not obvious. Surprisingly, there is an elegant solution to this problem with a good performance.

Instead of removing symbols we repeat them random number of times and we introduce correlation between the random decisions so that our process achieves synchronization. Indeed, deleting symbols would not work for us as we want to preserve the overall information contained in the strings. Although the underlying mechanism of our algorithm and that of [Sah14] is different, the analysis in [Sah14] eventually reduces to a similar problem about random walks. One of an open problems about the oblivious synchronization is whether one can design such a mechanism with a better performance than ours.

1.2 Related Works

The notion of edit distance plays a central role in several domains such as computational biology, speech recognition, text processing and information retrieval. As a consequence, computational problems involving edit distance seek attentions of many researchers. We refer the reader to the excellent survey by Navarro [Nav01] for a comprehensive treatment of this topic. The problem of computing exact edit distance can be solved in $O(n^2)$ time using classical dynamic programming based algorithm [WF74]. Later, Masek and Paterson [MP80] achieved slightly improved bound of $O(n^2/\log^2 n)$ on this problem and this is the best known upper bound so far. For the decision version of this problem, an $O(n+k^2)$ time algorithm is known [LMS98], where k is the edit distance between two input strings. On the other hand, if we focus on approximating edit distance, we have much better bounds on running time. The exact algorithm given in [LMS98] immediately gives a linear-time $\sqrt{n}$-approximation algorithm. Bar-Yossef *et al.* [BYJKK04] improved this approximation factor to $n^{3/7}$ with a (nearly) linear-time algorithm and later it was improved to $n^{1/3+o(1)}$ [BES06] and finally to $2^{\widetilde{O}(\sqrt{\log n})}$ [AO09]. The improved approximation factor by [AO09] was based on embedding edit distance into Hamming distance [OR07], where authors showed such an embedding

with distortion factor $2^{O(\sqrt{\log n \log\log n})}$. In case of embedding of *edit distance with moves* [2] into Hamming metric, the distortion factor is known to be upper bounded by $O(\log n \log^* n)$ [CM02]. Andoni *et al.* [ADG+03] showed a lower bound of 3/2 on distortion factor in case of embedding from edit distance metric to Hamming metric.

One of the implications of such randomized embedding is the document exchange problem. If we focus only on one-way communication protocol, then we have an upper bound of $O(k \log n)$ on the total number of bits transmitted [Orl91], but unfortunately that protocol incurs Bob's running time exponential in k. Jowhari [Jow12] studied this problem with the restriction that both Alice and Bob run $poly(k, n)$-time algorithms and gave a protocol with a $O(k \log^2 n \log^* n)$ bound on the number of bits transmitted. One of the main contributions of his work was to provide a time-efficient randomized embedding from edit distance metric into Hamming metric with distortion factor $O(\log n \log^* n)$ equipped with a polynomial time *reconstruction procedure*. At the end, he raised the question whether it is possible to achieve distortion factor $o(\log n)$ keeping both the embedding and the reconstruction procedure time-efficient. In this paper, we addressed this question and achieve an upper bound of k on distortion factor which is a significant improvement over the previous result when k is "small" compared to n. This in turns translates into a protocol with a $O(k^2 \log n)$ bound on the number of bits transmitted.

Another problem that we consider in this paper is the design of sketching algorithm to decide edit distance. For Hamming distance, efficient sketching algorithm is known [KOR98, BYJKK04] that solves the k vs. $(1+\epsilon)k$ *gap Hamming distance* problem with constant size sketches. Building on that result, Bar-Yossef *et al.* [BYJKK04] gave a computation of constant size sketch that can distinguish the two cases when edit distance is at most k and when that is at least $(kn)^{2/3}$, for $k \leq \sqrt{n}$. Later, improvement on distortion factor of embedding [OR07] results in solution to k vs. $2^{O(\sqrt{\log n \log\log n})} \cdot k$ gap edit distance problem. Our embedding result can be used to solve k vs. ck^2 gap edit distance problem for some constant c with constant size sketches.

1.3 Applications of the Main Result

As a consequence of Theorem 1.1, we achieve a better bound on number of bits transmitted in a one-way protocol solving the document exchange problem. The most obvious protocol in this regard is the following [Jow12]: Alice and Bob first compute $f(x, r)$ and $f(y, r)$, where r is the shared randomness and then they run the one-way protocol for document exchange problem under Hamming metric. We use the following lemma from [PL07] which provides an efficient sketching algorithm in case of Hamming distance.

Lemma 1.3 ([PL07]). *For two strings $x, y \in \{0,1\}^n$ such that $\Delta_H(x, y) \leq h$, there exists a randomized algorithm that maintains sketches of size $O(h \log n)$ and using sketches $s_h(x)$ and $s_h(y)$, it outputs all the tuples $\{(x_i, y_i)\}$ where $x_i \neq y_i$ with probability at least $(1 - 1/n)$ in $O(h \log n)$ time. Construction of sketch $s_h(x)$ can be done in $O(n \log n)$ time and space in one pass.*

Now if $\Delta_e(x, y) \leq k$, Bob will learn $f(x, r)$ and then using decoding algorithm he can get back x. After having x, Bob can decide $\Delta_e(x, y) \leq k$ in $O(n + k^2)$ time using the algorithm by [LMS98]. This idea leads us to the following corollary.

Corollary 1.4. *In the two-party communication model, there is a randomized one-way protocol that solves document exchange problem with high probability while transmitting only $O(k^2 \log n)$ bits. The running time of each party will be $O(n \log n + k^2 \log n)$.*

[2] Similar to $\Delta_e(x, y)$ with addition of a block move operation, where moving a substring of x to another location is considered as a single operation.

Another straightforward but important application of Theorem 1.1 is that it provides us a randomized sketching algorithm for k vs. ck^2 gap edit distance problem for some constant c. For this purpose, we need the following lemma from [BYJKK04].

Lemma 1.5 ([BYJKK04]). *For any $\epsilon > 0$ and k, k vs. $(1+\epsilon)k$ gap Hamming distance problem can be solved using an efficient sketching algorithm that maintains sketches of size $O(1/\epsilon^2)$ and if the set of non-zero coordinates of each input string can be computed in time t, then running time of Alice and Bob will be bounded by $O(\epsilon^{-3} t \log n)$.*

Now given two input strings x and y, we can first use embedding f of Theorem 1.1 and then apply the above lemma to get the following corollary.

Corollary 1.6. *There exists a $c \in \mathbb{N}$ such that for any k, there is a randomized sketching algorithm that solves k vs. ck^2 gap edit distance problem with high probability using sketches of size $O(1)$ attaining an upper bound of $O(n \log n)$ on Alice and Bob's running time.*

Among other implications of embedding, one interesting problem is to design *approximate nearest neighbor search* algorithms which is defined as given a database of m points, we have to pre-process such that given a query point, it would be possible to efficiently find a database point close to the query point. For Hamming metric, a search algorithm is known [IM98] that retrieves a database point which is at most $(1+\epsilon)$ times far from the closest one. Together with that, our embedding result implies a randomized algorithm that will return a point (under edit distance metric) within the distance of $O(k)$ times that of the closest one.

2 Preliminaries

In the rest of the paper, we refer to a random walk on a line as a random walk on the integer line where in each step with probability 1/2 we stay at the same place, with probability 1/4 we step to the left, and otherwise we step to the right. For parameters $t \in \mathbb{N}, \ell \in \mathbb{Z}$, We denote by $q(t,\ell)$ the probability that a random walk on a line starting at the origin reaches the point ℓ at time t for the first time (for convenience we set $q(0,0) = 1$).

Observation 2.1. *Let $t \in \mathbb{N}$ then:*

1. *For all $\ell < 0$ it holds that $q(t,\ell) \le q(t,\ell+1)$, and for all $\ell > 0$, $q(t,\ell) \le q(t,\ell-1)$,*
2. *For all $\ell \in \mathbb{N}$ it holds that $q(t,\ell) = \frac{1}{4}q(t-1,\ell-1) + \frac{1}{2}q(t-1,\ell) + \frac{1}{4}q(t-1,\ell+1)$,*
3. *For all $\ell > 1$ it holds that $q(t,\ell) = \sum_{j<t} q(t-j,\ell-1)q(j,1)$.*

The following is a well known fact about random walks that can be found e.g. in [LPW06, Theorem 2.17].

Proposition 2.2 (Folklore). *For any $k, \ell \in \mathbb{N}$ it holds that:*

$$\sum_{t=0}^{\ell} q(t,k) \ge 1 - \frac{12k}{\sqrt{\ell}}.$$

In particular, $\sum_{t=0}^{1296k^2} q(t,k) \ge \frac{2}{3}$.

3 The Basic Embedding

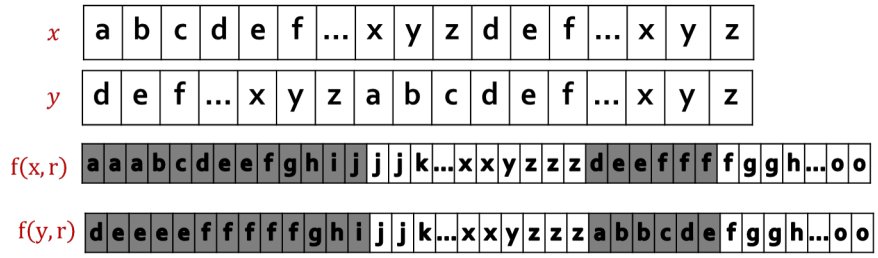
In this section we present our basic embedding and in the subsequent section we show how to reduce the number of its random bits to prove our main theorems. The pseudo-code for the embedding is given in Algorithm 1.

Algorithm 1 Basic Embedding Function f

Input : $x \in \{0,1\}^n$, and a random string $r \in \{0,1\}^{6n}$
Output: $f(x,r) \in \{0,1\}^{3n}$
Interpret r as a description of $h_1, \ldots, h_{3n} : \{0,1\} \to \{0,1\}$.
Initialization: $i = 1$, $Output = \lambda$;
for $j = 1, 2, \ldots, 3n$ **do**
  if $i \leq n$ **then**
    $Output = Output \odot x_i$, where the operation $\odot$ denotes concatenation;
    $i = i + h_j(x_i)$;
  end
  else
    $Output = Output \odot 0$;
  end
end
Set $f(x,r) = Output$.

Let us illustrate our embedding applied on the strings: $x = abc, \ldots, xyz, def, \ldots, xyz$ and $y = def, \ldots, xyz, abc, \ldots, xyz$ while using the same sequence r as random string.

Figure 1: Runtime example of the embedding algorithm:



As one can see, upon each block of edit changes, the output strings $f(x,r)$ and $f(y,r)$ become different, till they "synchronize" again. In the sequel, we justify this kind of behavior and show that the synchronization is rapid.

We summarize here the main properties of our basic embedding:

Theorem 3.1. *The mapping $f : \{0,1\}^n \times \{0,1\}^{6n} \to \{0,1\}^{3n}$ computed by Algorithm 1 satisfies the following conditions:*

1. *For every $x \in \{0,1\}^n$, given $f(x,r)$ and r, it is possible to decode back x with probability $1 - \exp(-\Omega(n))$.*
2. *For every $x, y \in \{0,1\}^n$, $\Delta_e(x,y)/2 \leq \Delta_H(f(x,r), f(y,r))$ with probability at least $1 - \exp(-\Omega(n))$.*

3. *For every positive constant c and every $x, y \in \{0,1\}^n$, $\Delta_H(f(x,r), f(y,r)) \leq c \cdot (\Delta_e(x,y))^2$ with probability at least $1 - \frac{12}{\sqrt{c}}$.*

Moreover, both the mapping f and its decoding (given $f(x,r)$ and r) take linear time and can be performed in a streaming fashion.

Let us justify the properties of our basic embedding from Theorem 3.1. It is clear from the algorithm that $|f(x,r)| = 3n$. For the first property observe that we can recover x from $f(x,r)$ and r provided that $i = n+1$ at the end of the run of the algorithm. Since at each iteration of the algorithm, i is incremented with probability $1/2$, the probability that during $3n$ rounds it does not reach $n+1$ can be bounded by $2^{-\Omega(n)}$ by the Chernoff bound. So unless this low probability event happens we can recover x from $f(x,r)$ and r.

Proving the second property is straightforward. Indeed, let $k = \Delta_e(x,y)$. We claim that $k/2 \leq \Delta_H(f(x,r), f(y,r))$ whenever the algorithm ends with $i = n+1$ on both x and y. In such a case x can be obtained from $f(x,r)$ by removing all the bits where $h_j(f(x,r)_j) = 0$. Similarly for y. Hence, y differs from x only in the part which is obtained from the portion of $f(y,r)$ which bit-wise differs from $f(x,r)$. If $\ell = \Delta_H(f(x,r), f(y,r))$ then we need to apply at most ℓ edit operations on x to obtain all the y except for at most the last ℓ bits of y (in the case when they are all 0). So except for an event that happens with exponentially small probability $\Delta_e(x,y) \leq 2 \cdot \Delta_H(f(x,r), f(y,r))$.

The rest of this section is devoted for the proof of Property 3. We will need the following main technical lemma. Together with Proposition 2.2 it implies the theorem.

Lemma 3.2. *Let $x, y \in \{0,1\}^n$ be of edit distance $\Delta_e(x,y) = k$. Let $q(t,k)$ be the probability that a random walk on the integer line starting from the origin visits the point k at time t for the first time. Then for any $\ell > 0$, $\Pr[\Delta_H(f(x,r), f(y,r)) \leq \ell] \geq \sum_{t=0}^{\ell} q(t,k)$ where the probability is over the choice of r.*

Consider two strings $x, y \in \{0,1\}^n$ such that $\Delta_e(x,y) = k$. We will analyze the behavior of the embedding function on these two strings. We are interested in the Hamming distance of the output of the function. Let $y = x^{(0)}, x^{(1)}, \ldots, x^{(k)} = x$ be a series of strings such that $\Delta_e(x^{(\ell-1)}, x^{(\ell)}) = 1$. Such strings exist by our assumption on the edit distance of x and y. Let i_ℓ be the first index on which $x^{(\ell-1)}$ and $x^{(\ell)}$ differ. Without loss of generality assume $i_1 < \cdots < i_k$. For a fixed value of $h_1, \ldots, h_{3n}$ we define a function $i_x : [3n] \to [n]$ such that $i_x(j)$ is the value of i in the j-th iteration of Algorithm 1 applied on x.

Let d_j measure the difference between deleted and inserted bits between x and y that were seen till iteration j, i.e. d_0 is initialized to 0, and d_j is defined recursively as follows: whenever j is such that $i_x(j) \notin \{i_1, \ldots, i_k\}$ then $d_j = d_{j-1}$. Otherwise, suppose $i_x(j) = i_\ell$, then $d_j = d_{j-1} - 1$ when $x^{(\ell+1)}$ is obtained by deletion from $x^{(\ell)}$, it is d_{j-1} when it was a bit flip and $d_{j-1} + 1$ when it was a insertion. The main observation is as follows:

Observation 3.3. *Let $j \in [3n]$ be such that $i_x(j) \in [i_\ell, i_{\ell+1})$, then:*

1. *If $i_x(j) = i_y(j) + d_j$, then $x_{i_x(j)} = y_{i_y(j)}$ so $f(x,r)_j = f(y,r)_j$.*
2. *Moreover if $i_x(j) = i_y(j) + d_j$, for every $j' \geq j$ if $i_x(j') < i_{\ell+1}$, then $i_x(j') = i_y(j') + d_j$. Overall, for every $j' \geq j$ satisfying $i_x(j') < i_{\ell+1}$ it holds that $f(x,r)_{j'} = f(y,r)_{j'}$.*

The first item follows easily by the definition of d_j and $x^{(0)}, \ldots, x^{(k)}$. As for the second item, observe that as long as $i_x(j') \leq i_{\ell+1}$ the increment of $i_x(j'), i_y(j')$ is identical as $x_{i_x(j')} = y_{i_y(j')}$, so in particular $h_{j'}(x_{i_x(j')}) = h_{j'}(y_{i_y(j')})$.

To bound $\Delta_H(f(x,r), f(y,r))$ we define the following randomized process which is induced by the behavior of the algorithm on x and y. The process consists of a particle moving randomly on the integer line and a goalspot. In the beginning both the particle and the goalspot are located at the origin. The process lasts $3n$ units of time.

The goal spot is moved according to d_j, i.e. whenever $i_x(j)$ hits an index i_ℓ, then the goalpost is moved according to the following rule:

- If $x^{(\ell)}$ is obtained from $x^{(\ell-1)}$ by a bit flip then the goalspot remains in its place.
- If $x^{(\ell)}$ is obtained from $x^{(\ell-1)}$ by a bit insertion then the goalspot shifts one step to the right.
- If $x^{(\ell)}$ is obtained from $x^{(\ell-1)}$ by a bit deletion then the goalspot shifts one step to the left.

The particle moves according to the following rule: If j is such that $f(x,r)_j = f(y,r)_j$ then the particle is idle. Otherwise, it makes a step according to $h_j(x_{i_x(j)}) - h_j(y_{i_y(j)})$. Clearly, $\Delta_H(f(x,r), f(y,r))$ equals the number of steps in which the particle is not idle (including the steps it remains in its place). Observe that whenever $f(x,r)_j \neq f(y,r)_j$ then $x_{i_x(j)} \neq y_{i_y(j)}$. Therefore, in such a case, since h_j is random, the particle shifts to the left/right with probability $1/4$ and stays in its place with probability $1/2$.

Let $j \in [3n]$ satisfying $i_\ell \leq i_x(j) < i_{\ell+1}$. The goalpost position in iteration j measures the difference between deleted and inserted bits in the series $x^{(1)}, \ldots, x^{(\ell)}$, namely it equals d_j. The particle position measures the difference between the increments of $i_x(j)$ and $i_y(j)$. Therefore, if the particle reaches the goalspot position then by Observation 3.3 it holds $x_{i_x(j)} = y_{i_y(j)}$. And by the second item of Observation 3.3 this implies that the particle would stay idle till the next iteration in which $i_x(j) = i_{\ell+1}$. Therefore, we need to analyze how many steps the particle performs after it becomes non-idle till it reaches the goalspot again. For this sake we define a new process that is easier to analyze:

The Kennel Struggle: Let us consider a process involving a dog, a cat and a kennel. All the involving entities are located on the integer line. In the beginning the kennel is located on the origin and so are the dog and the cat. The dog would like the cat to step out of the kennel. To this end the dog can perform one of the following actions:

The dog can bark, forcing the cat to perform a random step (defined shortly). Alternatively, the dog can move with kennel one step towards his preferred side. Whenever the cat is in the kennel the dog must perform an action. If the cat is not in the kennel, the dog may perform an action or stay idle. The dog's decision is based only on the cat position. Upon performing k actions the dog gives up and vanishes so the kennel is empty from then on (where $k = \Delta_e(x,y)$).

The cat, upon each dog barking, or whenever she is not at the kennel performs a random step: she steps to the left/right with probability $1/4$ and stays in its place with probability $1/2$. If the cat finds the kennel empty, then she happily stays there and the game is over.

It can be easily seen that for each configuration of the particle and goalpost process, there is a strategy for the dog such that: The distribution of the cat steps equals to the distribution of the particle moves, with the little change that we do not stop the particle and goalpost process after $3n$ steps (in the kennel struggle we skip the idle steps). Observe that if we do not end the particle and goalpost process after $3n$ steps the number of steps made by the particle is just larger. Therefore, an upper bound on the number of cat steps under any dog strategy, translates into an upper bound on the number of particle non-idle steps, which in turn bounds the Hamming distance of x and y.

Fix a strategy S for the dog, and denote by $p_S(\ell)$ the probability that after at most ℓ steps the cat reaches an empty kennel, provided that the dog acts according to S. The following claim implies Lemma 3.2.

Lemma 3.4. *Let $k \in \mathbb{N}$. Consider the kennel struggle, where the dog performs k actions. For every strategy S of the dog, $p_S(\ell) \geq \sum_{t=0}^{\ell} q(t,k)$.*

Proof. The lemma is a consequence of the following claim.

Claim 3.5. *The following dog's strategy $\mathcal{S}$ minimizes $p_{\mathcal{S}}(t)$ for every t: Wait till the cat reaches the kennel and then push the kennel to the right.*

Let us conclude the proof using the claim. Consider the dog's strategy given by Claim 3.5. In this strategy the probability $p_{\mathcal{S}}(\ell)$ is given by:

$$\sum_{t_1,\ldots,t_k | t_1+\cdots+t_k \leq \ell} (q(t_1,1)\cdots q(t_k,1)) = \sum_{t=0}^{\ell} \sum_{t_1,\ldots,t_k | t_1+\cdots+t_k = t} (q(t_1,1)\cdots q(t_k,1)). \tag{1}$$

On the other hand, the inner sum in (1) equals the probability that a random walk on line starting at the origin reaches place k at time t. To see that, observe that the last event can be phrased as follows: First compute the probability that the walk reaches place 1 in t_1 steps. Conditioned on that compute the probability that it reaches 2 in t_2 steps, and so on. Clearly, these events are independent. In order to get a total number of t steps we require $t_1 + \cdots + t_2 = t$.

Let us put things together. If the dog acts according to the strategy given by Claim 3.5 (which minimizes $p_S(\ell)$ for every ℓ), then for all values of $\ell > 0$: $p_{\mathcal{S}}(\ell) = \sum_{t=0}^{\ell} q(t,k)$. The lemma follows.

Proof of Claim 3.5. Consider a best strategy for the dog. That is, a series of decisions conditioned on the current position of the cat. We divide the dog's strategy into intervals, where each interval lasts until the cat reaches the kennel. Observe first that the distribution on the number of steps made by the cat in each interval is independent on the other intervals. Therefore in order to conclude the claim we show that in each interval separately we can replace the dog action by waiting till the cat reaches the kennel and then pushing it to the right.

Fix such an interval. Consider the last action in this interval, we first show that it must be a push of the kennel further from the cat. Let d be the distance of the cat from the kennel before the dog pushes the kennel and let $t \in \mathbb{N}$. We claim that for any value of t the probability that the cat reaches an empty kennel within t steps is minimized when the dog pushes the kennel further from the cat. Indeed:

- If the dog chooses to bark, then the above probability is given by: $\frac{1}{4}q(t,d-1) + \frac{1}{2}q(t,d) + \frac{1}{4}q(t,d+1) = q(t+1,d)$.
- If the dog chooses to push the kennel towards the cat, then the above probability is given by:$\frac{1}{4}q(t,d-2) + \frac{1}{2}q(t,d-1) + \frac{1}{4}q(t,d) = q(t+1,d-1)$.
- If the dog chooses to push the kennel further from the cat, then the above probability is given by: $\frac{1}{4}q(t,d) + \frac{1}{2}q(t,d+1) + \frac{1}{4}q(t,d+2) = q(t+1,d+1)$.

The equality in each case is obtained by Observation 2.1 Item 2. By Item 1 of Observation 2.1 we conclude that the probability that the cat reaches an empty kennel within t steps is minimized when the dog pushes the kennel further from the cat.

Now we show that if we replace the last action by waiting until the cat reaches the kennel, and then pushing it to the right, the distribution on the cat steps does not change. The probability that the cat reaches the kennel in t steps is given by: $q(t,d+1)$. If the dog instead waits until the cat

reaches the kennel and then pushes the kennel to the right. Then probability that the cat reaches (again) the kennel in t steps is given by: $\sum_{i=1}^{t}(q(i,d))q(t-i,1)$ which, by Observation 2.1, equals $q(t,d+1)$, the claim follows. In such a way, it can be shown that each kennel push in this interval, can be be replaced by waiting till the cat reaches the kennel and then pushing the kennel to the right. □

□

4 Reducing the randomness

In this section we show how to reduce the number of random bits used by Algorithm 1 to derive Theorems 1.1 and 1.2.

An easy observation is that for the Boolean case, one can save a half of the random bits needed for our basic embedding function given by Algorithm 1. We may replace the current condition for incrementing i that $h_j(x_i) = 1$ by the condition $r_j = x_i$. As one can verify in our analysis of the third property in Theorem 3.1 this would not substantially affect the property of the algorithm. It would actually improve the bound on the resulting Hamming distance by a factor of roughly 2 because the induced random walk would be non-lazy. To obtain more substantial savings we will use tools from derandomization.

By standard probabilistic method similar to [Gol01, Proposition 3.2.3], we argue that there exists a subset of $\{0,1\}^{6n}$ of substantially *small* size (of size at most $2n^{c'}$), from which sampling r is "almost" as good as sampling it uniformly from $\{0,1\}^{6n}$. Thus by hard-wiring R inside the algorithm and sampling r from R, we get the desired non-uniform algorithm promised in Theorem 1.1, details below.

By choosing c appropriately and for large enough n, our basic embedding function f has the property that with probability at least $3/4$, for random $r \in \{0,1\}^{6n}$

$$\frac{1}{2} \cdot \Delta_e(x,y) \leq \Delta_H(f(x,r), f(y,r)) \leq c \cdot (\Delta_e(x,y))^2.$$

Now take a random subset $R \subseteq \{0,1\}^{6n}$ of the size of the smallest power of two $\geq n^{c'}$, for some suitable constant $c' > 0$. Fix $x, y \in \{0,1\}^n$. By Chernoff bound, the probability that

$$\left| \Pr_{r\in\{0,1\}^{6n}}\left[\frac{1}{2} \cdot \Delta_e(x,y) \leq \Delta_H(f(x,r), f(y,r)) \leq c \cdot (\Delta_e(x,y))^2\right] - \Pr_{r\in R}\left[\frac{1}{2} \cdot \Delta_e(x,y) \leq \Delta_H(f(x,r), f(y,r)) \leq c \cdot (\Delta_e(x,y))^2\right] \right| > \frac{1}{n}$$

over the random choice of R is at most 2^{-2n}. Hence, by the union bound over all x and y, there is a set R of the required size such that for any $x, y \in \{0,1\}^n$,

$$\left| \Pr_{r\in\{0,1\}^{6n}}\left[\frac{1}{2} \cdot \Delta_e(x,y) \leq \Delta_H(f(x,r), f(y,r)) \leq c \cdot (\Delta_e(x,y))^2\right] - \Pr_{r\in R}\left[\frac{1}{2} \cdot \Delta_e(x,y) \leq \Delta_H(f(x,r), f(y,r)) \leq c \cdot (\Delta_e(x,y))^2\right] \right| \leq \frac{1}{n}.$$

Thus instead of sampling r from the whole universe $\{0,1\}^{6n}$ we can sample r from R without affecting the probability of small distortion by more than $1/n$. Since R is of size at most $2n^{c'}$, we need only $\log|R| = \lceil c' \log n \rceil$ random bits to sample a random element of R. The non-uniform algorithm for the embedding function of Theorem 1.1 has R hard-wired as a table. On input x and $s \in \{0,1\}^{\log|R|}$ the algorithm simulates Algorithm 1 on x and the s-th string in R. By properties of R we know that such an algorithm satisfies the conclusion of Theorem 1.1.

The above algorithm has an optimal seed length but it has the disadvantage of storing a large table of non-uniformly selected strings (the subset R). To get rid of the table we will use Nisan's pseudo-random generator [Nis90].

Nisan's pseudo-random generator $G_{k,w}$ is a function that takes a seed s of length w and k pair-wise independent hash functions $h_1, h_2, \ldots, h_k : \{0,1\}^w \to \{0,1\}^w$ and outputs a string $r \in \{0,1\}^{w2^k}$ defined recursively as follows:

$$G_{0,w}(s) = s$$
$$G_{k,w}(s, h_1, h_2, \ldots, h_k) = G_{k-1,w}(s, h_1, \ldots, h_{k-1}) \odot G_{k-1,w}(h_k(s), h_1, \ldots, h_{k-1})$$

Nisan proved that his generator satisfies the following property.

Theorem 4.1 ([Nis90]). *For an arbitrary constant $c_0 > 0$, let A be an algorithm that uses work space of size at most $c_0 \log n$ and runs in time at most n^{c_0} with a two-way access to its input string x and a one-way access to a random string r. There is a constant $c_1 > 0$ such that*

$$\left| \Pr_{r}[A(x,r) \text{ accepts}] - \Pr_{r'}[A(x,r') \text{ accepts}] \right| < \frac{1}{n^2}$$

where r is taken uniformly at random, and r' is taken according to the distribution induced by $G_{c_0 \log n, w}(s, h_1, \ldots, h_{c_0 \log n})$ where $w = c_1 \log n$, $s \in \{0,1\}^w$ is taken uniformly at random and each h_i is sampled independently from an ensemble of pair-wise independent hash functions.

There are ensembles of pair-wise independent hash functions mapping w bits into w bits where each function is identified by a binary string of length $O(w)$. Nisan [Nis90] gives several such examples and there are many others. In particular, the ensemble given by Dietzfelbinger [Die96] can be evaluated on word RAM with word size $O(w)$ using $O(1)$ multiplications and bit operations.

We would like to apply Theorem 4.1 on Algorithm 1. However, Theorem 4.1 applies only for decision algorithms. Therefore we define the following algorithm A:

Algorithm 2 Hamming Distance Test

Input : $x, y \in \{0,1\}^n$, $k \in \{1, \ldots, 3n\}$, and a random string $r \in \{0,1\}^{6n}$
Output: Accept iff: $\Delta_H(f(x,r), f(y,r)) = k$
Compute $\Delta_H(f(x,r), f(y,r))$ for the basic embedding function f by simultaneously computing $f(x,r)$ and $f(y,r)$ while counting the Hamming distance of $f(x,r)$ and $f(y,r)$;
Accept if $\Delta_H(f(x,r), f(y,r)) = k$;

Given the properties of our Algorithm 1 for the basic embedding function, it is clear that A processes its input in logarithmic space using one-way access to its random string r. Hence, we can apply Theorem 4.1 on algorithm A. That implies that the distributions of the Hamming distance

$\Delta_H(f(x,r), f(y,r))$ on a random string r and a random string r' sampled according to Nisan's pseudo-random generator are close in ℓ_∞ distance.

Hence, instead of providing Algorithm 1 with completely random string we will provide it with a seed of length $O(\log n)$ and a sequence $O(\log n)$ of hash functions that will be expanded by the Nisan's pseudo-random generator into a full pseudo-random string r'. This r' is used in place of r to compute $f(x,r')$. Since each hash function can be specified using $O(\log n)$ bits this algorithm will require only $O(\log^2 n)$ random bits in total.

Furthermore, it is clear from the description of the Nisan's pseudo-random generator, that each bit of r' can be obtained by evaluating at most $O(\log n)$ hash functions. When computing r' bit by bit we only need to evaluate $O(1)$ hash function on average. Thus when using Dietzfelbinger's hash functions on a RAM with word size $O(\log n)$ we can compute $f(x,r')$ in a streaming fashion spending only $O(1)$ operations per output bit on average and $O(\log n)$ in the worst-case. This proves Theorem 1.2.

5 Non-binary alphabets

Our results carry over directly to larger alphabets of constant size. For alphabets Σ_n where the size of Σ_n depends on n we assume that the symbols are binary encoded by strings of length $\log|\Sigma_n|$. Our basic embedding given by Algorithm 1 only needs that each $h_1, \ldots, h_{3n}$ is a pair-wise independent hash function from Σ_n to $\{0,1\}$. Such a hash function is obtained for example by selecting a random vector $r \in \{0,1\}^{\log|\Sigma_n|}$ and a bit b, and taking the inner product of the binary encoding of an alphabet symbol with r and adding b over GF_2. Hence, one needs only $1+\log|\Sigma_n|$ bits to specify each hash function.

Thus Theorem 3.1 will use $n \cdot (1+\log|\Sigma_n|)$ random bits, Theorem 1.1 will have $\ell = O(\log n + \log\log|\Sigma_n|)$ and Theorem 1.2 will have $\ell = O((\log n + \log\log|\Sigma_n|)^2)$.

Acknowledgements

The authors thank anonymous reviewers for pointing out [Sah14, DW112] and other useful comments.

References

[ADG+03] Alexandr Andoni, Michel Deza, Anupam Gupta, Piotr Indyk, and Sofya Raskhodnikova, *Lower bounds for embedding edit distance into normed spaces.*, SODA, ACM/SIAM, 2003, pp. 523–526.

[AO09] Alexandr Andoni and Krzysztof Onak, *Approximating edit distance in near-linear time*, Proceedings of the Forty-first Annual ACM Symposium on Theory of Computing (New York, NY, USA), STOC '09, ACM, 2009, pp. 199–204.

[BES06] Tuğkan Batu, Funda Ergun, and Cenk Sahinalp, *Oblivious string embeddings and edit distance approximations*, Proceedings of the Seventeenth Annual ACM-SIAM Symposium on Discrete Algorithm (Philadelphia, PA, USA), SODA '06, Society for Industrial and Applied Mathematics, 2006, pp. 792–801.

[BYJKK04] Z. Bar-Yossef, T.S. Jayram, R. Krauthgamer, and R. Kumar, *Approximating edit distance efficiently*, Foundations of Computer Science, 2004. Proceedings. 45th Annual IEEE Symposium on, Oct 2004, pp. 550–559.

[CM02] Graham Cormode and S. Muthukrishnan, *The string edit distance matching problem with moves*, Proceedings of the Thirteenth Annual ACM-SIAM Symposium on Discrete Algorithms, January 6-8, 2002, San Francisco, CA, USA., 2002, pp. 667–676.

[CPSV00] Graham Cormode, Mike Paterson, Süleyman Cenk Sahinalp, and Uzi Vishkin, *Communication complexity of document exchange*, Proceedings of the Eleventh Annual ACM-SIAM Symposium on Discrete Algorithms (Philadelphia, PA, USA), SODA '00, Society for Industrial and Applied Mathematics, 2000, pp. 197–206.

[Die96] Martin Dietzfelbinger, *Universal hashing and k-wise independent random variables via integer arithmetic without primes*, STACS 96, 13th Annual Symposium on Theoretical Aspects of Computer Science, Grenoble, France, February 22-24, 1996, Proceedings, 1996, pp. 569–580.

[DW112] *Dortmund workshop on algorithms for data streams 2012*, 2012.

[Gol01] Oded Goldreich, *The foundations of cryptography - volume 1, basic techniques*, Cambridge University Press, 2001.

[IM98] Piotr Indyk and Rajeev Motwani, *Approximate nearest neighbors: Towards removing the curse of dimensionality*, Proceedings of the Thirtieth Annual ACM Symposium on Theory of Computing (New York, NY, USA), STOC '98, ACM, 1998, pp. 604–613.

[Jow12] Hossein Jowhari, *Efficient communication protocols for deciding edit distance*, Algorithms - ESA 2012 - 20th Annual European Symposium, Ljubljana, Slovenia, September 10-12, 2012. Proceedings, 2012, pp. 648–658.

[KN97] Eyal Kushilevitz and Noam Nisan, *Communication complexity*, Cambridge University Press, New York, NY, USA, 1997.

[KOR98] Eyal Kushilevitz, Rafail Ostrovsky, and Yuval Rabani, *Efficient search for approximate nearest neighbor in high dimensional spaces*, Proceedings of the Thirtieth Annual ACM Symposium on Theory of Computing (New York, NY, USA), STOC '98, ACM, 1998, pp. 614–623.

[Lev66] VI Levenshtein, *Binary Codes Capable of Correcting Deletions, Insertions and Reversals*, Soviet Physics Doklady **10** (1966), 707.

[LMS98] Gad M. Landau, Eugene W. Myers, and Jeanette P. Schmidt, *Incremental string comparison*, SIAM J. Comput. **27** (1998), no. 2, 557–582.

[LPW06] David A. Levin, Yuval Peres, and Elizabeth L. Wilmer, *Markov chains and mixing times*, American Mathematical Society, 2006.

[MP80] William J. Masek and Michael S. Paterson, *A faster algorithm computing string edit distances*, Journal of Computer and System Sciences **20** (1980), no. 1, 18 – 31.

[Nav01] Gonzalo Navarro, *A guided tour to approximate string matching*, ACM Comput. Surv. **33** (2001), no. 1, 31–88.

[Nis90] N. Nisan, *Pseudorandom generators for space-bounded computations*, Proceedings of the Twenty-second Annual ACM Symposium on Theory of Computing (New York, NY, USA), STOC '90, ACM, 1990, pp. 204–212.

[OR07] Rafail Ostrovsky and Yuval Rabani, *Low distortion embeddings for edit distance*, J. ACM **54** (2007), no. 5, 23+.

[Orl91] Alon Orlitsky, *Interactive communication: Balanced distributions, correlated files, and average-case complexity*, FOCS, IEEE Computer Society, 1991, pp. 228–238.

[PL07] Ely Porat and Ohad Lipsky, *Improved sketching of hamming distance with error correcting.*, CPM (Bin Ma and Kaizhong Zhang, eds.), Lecture Notes in Computer Science, vol. 4580, Springer, 2007, pp. 173–182.

[Sah14] Barna Saha, *The dyck language edit distance problem in near-linear time*, 55th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2014, Philadelphia, PA, USA, October 18-21, 2014, 2014, pp. 611–620.

[WF74] Robert A. Wagner and Michael J. Fischer, *The string-to-string correction problem*, J. ACM **21** (1974), no. 1, 168–173.

ECCC ISSN 1433-8092

http://eccc.hpi-web.de