Constant Rate Isometric Embeddings of Hamming Metric into Edit Metric

Sudatta Bhattacharya
Faculty of Mathematics and Computer Science, Weizmann Institute of Science, Rehovot, Israel
Computer Science Institute of Charles University, Prague, Czech Republic

Sanjana Dey
UMONS – Université de Mons, Belgium

Elazar Goldenberg
The Academic College of Tel Aviv-Yaffo, Israel

Mursalin Habib
Rutgers University, Piscataway, NJ, USA

Bernhard Haeupler
INSAIT, Sofia University "St. Kliment Ohridski", Bulgaria
ETH Zürich, Switzerland

Karthik C. S.
Rutgers University, Piscataway, NJ, USA

Michal Koucký
Computer Science Institute of Charles University,Prague, Czech Republic

Abstract

A function $\varphi : \{0,1\}^n \to \{0,1\}^N$ is called an *isometric embedding* of the n-dimensional Hamming metric space to the N-dimensional edit metric space if, for all $x, y \in \{0,1\}^n$, the Hamming distance between x and y is equal to the edit distance between $\varphi(x)$ and $\varphi(y)$. The *rate* of such an embedding is defined as the ratio n/N.

It is well known in the literature how to construct isometric embeddings with a rate of $\Omega\left(\frac{1}{\log n}\right)$. However, achieving even near-isometric embeddings with a positive constant rate has remained elusive until now.

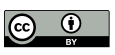
In this paper, we present an isometric embedding with a rate of $\frac{1}{8}$ by discovering connections to *synchronization strings*, which were studied in the context of insertion-deletion codes (Haeupler-Shahrasbi [JACM'21]). At a technical level, we introduce a framework for obtaining high-rate isometric embeddings using a novel object called a *misaligner*. We speculate that, with sufficient computational resources, our framework could potentially yield isometric embeddings with a rate of $\frac{1}{5}$.

As an immediate consequence of our constant rate isometric embedding, we improve known conditional lower bounds for the closest pair problem and the discrete 1-center problem in the edit metric and NP-hardness of approximation results for clustering problems and the Steiner tree problem in the edit metric, but now with optimal dependency on the dimension. Furthermore, we obtain optimal lower bounds for the gap edit distance problem in the two-player randomized communication complexity model.

We complement our results by showing that no isometric embedding $\varphi : \{0,1\}^n \to \{0,1\}^N$ can have rate greater than $\frac{15}{32}$ for all positive integers n. En route to proving this upper bound, we uncover fundamental structural properties necessary for every Hamming-to-edit isometric embedding. We also prove similar upper and lower bounds for embeddings over larger alphabets.

Finally, we consider embeddings $\varphi : \Sigma_{\text{in}}^n \to \Sigma_{\text{out}}^N$ between different input and output alphabets, where the rate is given by $\frac{n \log |\Sigma_{\text{in}}|}{N \log |\Sigma_{\text{out}}|}$. In this setting, we show that the rate can be made arbitrarily close to 1.

2012 ACM Subject Classification Theory of computation → Random projections and metric embeddings; Theory of computation → Problems, reductions and completeness; Theory of computation → Pattern matching



53rd International Colloquium on Automata, Languages, and Programming (ICALP 2026).
Editors: Sayan Bhattacharya, Danupon Nanongkai, Michael Benedikt, and Gabriele Puppis;
Article No. 32; pp. 32:1–32:19

Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Keywords and phrases Edit distance, Hamming distance, metric embeddings, synchronization strings, fine-grained complexity

Digital Object Identifier 10.4230/LIPIcs.ICALP.2026.32

Category Track A: Algorithms, Complexity and Games

Related Version *Full Version*: `https://arxiv.org/abs/2504.03605` [6]

Supplementary Material *Software (Source Code)*: `https://anonymous.4open.science/r/ham-to-edit-11D8`

Funding *Sudatta Bhattacharya*: Supported by the Grant Agency of the Czech Republic under grant agreement no. 24-10306S and by Charles University projects UNCE 24/SCI/008 and GAUK125424. Part of this work was carried out during a visit to DIMACS, with support from the National Science Foundation under Grant CCF-1836666.
Sanjana Dey: Partially supported by the Fonds de la Recherche Scientifique – FNRS under Grant no. T.0188.23 (PDR ControlleRS). Part of this work was carried out during a visit to DIMACS, with support from the National Science Foundation under Grant CCF-1836666.
Elazar Goldenberg: Partially supported by the National Science Foundation under Grant CCF-2313372. Part of this work was carried out during a visit to DIMACS, with support from the National Science Foundation under Grant CCF-1836666.
Mursalin Habib: Supported by the National Science Foundation under Grants CCF-2313372, CCF-2443697, by the Simons Foundation through Grant No. 825876, Awardee Thu D. Nguyen, and partially funded by the Ministry of Education and Science of Bulgaria's support for INSAIT as part of the Bulgarian National Roadmap for Research Infrastructure.
Bernhard Haeupler: Partially funded by the Ministry of Education and Science of Bulgaria's support for INSAIT as part of the Bulgarian National Roadmap for Research Infrastructure, and the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (Grant Agreement No. 949272).
Karthik C. S.: Supported by the National Science Foundation under Grants CCF-2313372, CCF-2443697, by the Simons Foundation through Grant No. 825876, Awardee Thu D. Nguyen, and partially funded by the Ministry of Education and Science of Bulgaria's support for INSAIT as part of the Bulgarian National Roadmap for Research Infrastructure.
Michal Koucký: Supported by the Grant Agency of the Czech Republic under grant agreement no. 24-10306S and by Charles University project UNCE 24/SCI/008. Part of this work was carried out during a visit to DIMACS, with support from the National Science Foundation under Grant CCF-1836666.

1 Introduction

Metric embeddings offer a powerful framework for formally comparing metric spaces by mapping points from a source space into a target space. The goal of such a mapping is to preserve pairwise distances faithfully (i.e., minimize *distortion*), thereby revealing structural and computational relationships between different notions of distance. These embeddings are particularly useful for understanding connections among fundamental metrics, such as those central to coding theory, sequence analysis, and computational geometry.

Among the most fundamental distance measures arising in stringology and related fields are the Hamming and the edit distances. The edit distance quantifies the minimum number of character insertions, deletions, and substitutions required to transform one string into another, while restricting these operations to only substitutions on equal-length strings yields

the Hamming distance. Because these metrics capture different notions of string similarity, both play central roles in diverse areas, including pattern matching, machine learning, and computational biology [32, 29]. Consequently, numerous classical computational problems – like finding the closest pair of strings in a set [30, 15] or identifying a representative center string for a dataset [39, 33, 28] – are widely studied using these metrics, underpinning even public bioinformatics services [1].

Given the fundamental nature of both metrics, understanding their relationship via embeddings is crucial. This paper systematically studies embeddings of the Hamming metric into the edit metric, exploring how the simpler substitution-only distance structure can be represented within the richer, insertion- and deletion-allowing space. Formally, we investigate functions $\varphi : \{0,1\}^n \to \{0,1\}^N$, where the domain $\{0,1\}^n$ is equipped with the Hamming distance $\Delta_{\mathsf{Hamming}}$ and the codomain $\{0,1\}^N$ with the edit distance Δ_{edit}. We define the rate of the embedding φ to be $\frac{n}{N}$. If φ keeps distances unchanged except possibly multiplying all of them by some fixed constant, i.e., for some universal constant $K \geq 1$, we have $\Delta_{\mathsf{edit}}\ (\varphi(x),\ \varphi(y)) = K\ \cdot\ \Delta_{\mathsf{Hamming}}(x,y)$, for all $x, y \in \{0,1\}^n$, then we call φ a 1-embedding. If $K = 1$, i.e., φ preserves distances exactly, then we call φ an *isometric embedding*.

The goal of this work is to seek answers to fundamental questions about these embeddings: Is it possible to achieve a positive constant rate 1-embedding, or better, an isometric embedding? What is the optimal rate achievable? Can we establish bounds on rates that are unattainable? Does using larger alphabets for isometric embeddings unlock the ability to obtain higher rates? By addressing all these questions, we aim to provide a comprehensive study of embeddings of the Hamming metric into the edit metric.

1.1 Quest for Constant Rate 1-Embedding

In computational complexity, it is typically easier to prove the intractability of problems in the Hamming metric, and in such a situation, if we had a 1-embedding of the Hamming metric into the edit metric, then the hardness result would translate to the edit metric as well. These embeddings have been studied in the literature for exactly this purpose. For instance, a folklore embedding[1] from the Hamming metric to the edit metric constructs the output string by inserting a uniformly random block of size $\Theta(\log n)$ after each character of the input string. This can be shown to result in an isometric embedding with high probability (and can be derandomized using constant relative distance codes in the edit metric, such as the ones in [36]).

However, this embedding achieves a rate of only $\frac{1}{\Theta(\log n)}$. Consequently, this implies that if a problem is hard for strings of length n in the Hamming metric, then it remains hard in the edit metric, but only for strings of length $\Theta(n \log n)$. On the other hand, our intuition suggests that solving problems in the n-dimensional edit metric should be at least as hard as solving the same problems in the n-dimensional Hamming metric. The inability to provide a formal justification for this intuition leaves us wanting for a clearer understanding. To address this gap, we aim to answer the following natural question:

Does there exist a 1-embedding of the Hamming metric into the edit metric with positive constant rate?

[1] See, for example, Lemma 11 in [2].

Our first result is an affirmative answer to the above question (in fact, we provide an isometric embedding). Our conceptual contribution in this regard is a clean connection between *synchronization strings* studied in the context of the study of codes for correcting insertion and deletion errors [19, 21, 17, 20, 10] with the above question.

▶ **Theorem 1.** *There exists a universal constant $C \geq 1$ such that for every positive integer n, there is an isometric embedding $\varphi_n : \{0,1\}^n \to \{0,1\}^{Cn}$ of the Hamming metric into the edit metric.*

This represents a sharp improvement: even for the weaker notion of *near-isometric* embeddings, the previously best known rate was only $\frac{1}{\Theta(\log n)}$ [35].

We remark here that although the proof of Theorem 1 is, in hindsight, simple and natural (see Section 2.1 for a proof sketch), the route to it was not obvious. Synchronization strings have been designed in literature to trade alphabet size for approximation accuracy, and had been regarded primarily as an approximation tool. It was not evident, at least to us, that they could be leveraged to obtain isometric embeddings.

As an immediate consequence of Theorem 1, we obtain the following meta theorem for *discrete* optimization problems in the edit metric. By discrete, we mean that the input is a set of points in the metric space, and the solution is some subset of the input points. For such problems, we have the following.

If a discrete optimization problem defined in the n-dimensional Hamming metric cannot be solved in time $T(n)$ (for some computable function T), then the same optimization problem defined in the n-dimensional Edit metric cannot be solved in time $T(\Theta(n))$.

As concrete manifestations of this meta theorem, we prove optimal hardness results in two distinct settings. We remark that in the case of all the three theorems below, i.e., Theorems 2, 3, and 6, the previously known lower bounds or hardness results in the edit metric over binary strings was only for dimensions $d = O(\log n \cdot \log\log n)$, where the earlier mentioned embedding with rate $\Theta\left(\frac{1}{\log d}\right)$ was used. Thus, by reducing the dimension to $d = O(\log n)$ in the three theorems below, we obtain optimal dependency in the dimension.

Fine-Grained Complexity

Using Theorem 1, we obtain conditional lower bounds for the closest pair problem and the 1-center problem in the edit metric with optimal dependency in the dimension.

▶ **Theorem 2.** *Unless the Strong Exponential Time Hypothesis is false, for every $\delta > 0$ there exists an $\varepsilon > 0$ such that given as input sets $A, B \subseteq \{0,1\}^d$ of N vectors (where $d = O_\delta(\log N)$), computing a $(1+\varepsilon)$-approximate closest pair in $A \times B$ in the edit metric requires time $\Omega(N^{2-\delta})$.*

The above theorem follows immediately by combining Theorem 1 with the conditional lower bound for the bichromatic closest pair problem in the Hamming metric obtained by [35]. Moreover, one can also obtain a conditional lower bound of $n^{1.5-o(1)}$ (with optimal dependency on dimension) against approximating the monochromatic closest pair problem in the edit metric by starting from [25]. Next, for the discrete 1-center problem, we have the following:

▶ **Theorem 3.** *Unless the Hitting Set Conjecture is false, for every $\varepsilon > 0$ there exists $c > 1$ such that given as input a point-set $P \subseteq \{0,1\}^d$ where $|P| = N$ and $d = c \log N$, computing the point $x \in P$ that minimizes the maximum edit distance to all points in P requires $\Omega(N^{2-\varepsilon})$ time.*

The above theorem follows immediately by combining Theorem 1 with the conditional lower bound for the discrete 1-center problem in the Hamming metric obtained by [2].

We also obtain improved lower bounds for data structures supporting dictionary look-ups and text indexing under the edit distance.

▶ **Theorem 4.** *Assuming the Strong Exponential Time Hypothesis, for every $\delta > 0$, there exists a constant c' such that any data structure with polynomial construction time that solves the dictionary look-up problem under the edit distance for a set of n binary strings of length $c' \log n$ cannot have query time $O(n^{1-\delta})$.*

▶ **Theorem 5.** *Assuming the Strong Exponential Hypothesis, for every $\delta > 0$, there exists a constant c' such that any data structure for text indexing under the edit distance for a text of length n, which can be constructed in polynomial time, cannot have query time $O(n^{1-\delta})$ even when pattern strings have length at most $c' \log n$.*

Theorems 4 and 5 follow directly by applying the embedding from Theorem 1 to the proofs of Corollaries 7 and 19 in [12], respectively.

NP Hardness

Applying Theorem 1 to known NP-hardness of approximation results for discrete clustering problems [13] and the discrete Steiner tree problem [14] in the Hamming metric, we obtain below their hardness of approximation result in the edit metric with optimal dependency in the dimension.

▶ **Theorem 6.** *It is NP-hard to approximate:*

- *the discrete k-means problem (resp. discrete k-center problem and discrete k-median problem) on N points in the $\{0,1\}^{O(\log N)}$ dimensional edit metric to a factor better than 1.38 (resp. $3 - o(1)$ and 1.12).*
- *the discrete Steiner tree problem on N terminals in the $\{0,1\}^{O(\log N)}$ dimensional edit metric to a factor better than 1.004.*

Beyond the meta-theorem for discrete optimization, the framework of transferring hardness via isometric embeddings extends to other computational models and complexity measures, including communication complexity, streaming algorithms, distributed algorithms, and online algorithms. As an illustration, we now discuss an application to communication complexity.

Communication Complexity

In the *Gap-Hamming problem*, Alice and Bob are each given n-bit strings, and the goal is to design a communication protocol that allows one party (say, Alice) to compute the Hamming distance between their strings to within $\pm\sqrt{n}$ using as little communication as possible. This problem was introduced by [22], and its complexity was settled by [8], who showed an $\Omega(n)$ randomized communication complexity lower bound (see also [38, 37, 16]). The Gap-Hamming problem is of particular interest in communication complexity as it has applications to proving lower bounds for many streaming algorithms, for example, frequency moment estimation [23] and entropy estimation [7].

We introduce a natural edit distance analogue: the *Gap-Edit problem* where Alice and Bob receive n-bit strings and must estimate their edit distance within $\pm\sqrt{n}$ using as little communication as possible. Applying Theorem 1 to the $\Omega(n)$ lower bound for the Gap-Hamming problem [8] immediately implies an identical $\Omega(n)$ randomized communication

complexity lower bound for the Gap-Edit problem. To the best of our knowledge, the Gap-Edit problem under this additive approximation has not been studied in the literature; a closely related multiplicative-approximation version has been investigated in [3].

1.2 High-Rate Isometric Embeddings

While Theorem 1 provides a constant-rate isometric embedding of the Hamming metric into the edit metric, the guaranteed rate is quite small. In certain natural applications, however, it is important that we have *high-rate* embeddings. Below, we present three concrete motivations for finding isometric embeddings that maximize the rate.

Motivation 1: The Exact-Edit Problem

Consider the following special case of the above mentioned Gap-Hamming problem (resp. Gap-Edit problem), named the *Exact-Hamming problem* (resp. *Exact-Edit problem*), where Alice and Bob are each given n-bit strings, and the goal is to design a communication protocol that lets one party compute the Hamming distance (resp. edit distance) between their strings *exactly*. The $\Omega(n)$ randomized communication complexity for the Gap-Hamming problem can be improved to $(1 - o_\varepsilon(1)) \cdot n$ for the Exact-Hamming problem[2], where ε is the error probability. This improvement is achieved by explicitly using the structure of the Hamming metric, and it is not clear how to extend it to the Exact-Edit problem. Directly applying Theorem 1 only yields that the randomized communication complexity for the Exact-Edit problem is $\Omega(n)$. Thus, to obtain a stronger lower bound, one seeks a high-rate isometric embedding.

Motivation 2: Codes in Edit Metric

Theorem 1 directly implies that good Hamming codes (those with positive constant rate and relative distance) yield corresponding codes in the edit metric. Since the isometric embedding given by the theorem simply involves interleaving the input bits with a sequence of fixed bit strings[3], these embedded codes preserve many structural properties of their Hamming counterparts. The drawback, however, is that the embedding reduces the code's rate and relative distance by the constant factor C introduced in the theorem statement.

Motivation 3: Hamming Cube in Edit Cube

Consider the weighted complete graph $\widetilde{Q}_n$ (resp. $\widetilde{A}_N$) on the vertex set $\{0,1\}^n$ (resp. $\{0,1\}^N$) where the weight of the edge on any two vertices corresponds to their Hamming distance (resp. edit distance). It is clear that for every pair of vertices, their weight in $\widetilde{A}_n$ is at most their weight in $\widetilde{Q}_n$. A natural question then arises: what is the smallest $N > n$, such that we can identify an isomorphic copy of $\widetilde{Q}_n$ in $\widetilde{A}_N$? This question of finding the largest Hamming cube that embeds isometrically into an edit cube is of pure combinatorial interest, independent of other applications.

---

[2] This improved lower bound is obtained by a reduction from the Inner Product problem, where Alice and Bob must compute the parity of the number of coordinates where both have a 1. The randomized communication complexity for the Inner Product problem with error probability $\frac{1}{2} - \varepsilon$ is at least $n - O(\log(1/\varepsilon))$ [11, 27]. Note that for every $x, y \in \{0,1\}^n$, we have $\sum_{i\in[n]} x_i \cdot y_i = (|x| + |y| - \Delta_{\mathsf{Hamming}}(x,y))/2$, and thus every protocol for the Exact-Hamming problem with t bits of communication implies a protocol for the Inner Product problem with $t + O(\log n)$ bits of communication.

[3] Such embeddings are referred to as interleaved embeddings throughout the paper; see Section 1.3.

The above three motivations drive us to find high-rate isometric embeddings of the Hamming metric into the edit metric leading to the following fundamental question:

What is the optimal rate for an isometric embedding of the Hamming metric into the Edit metric?

We make significant progress on this question and prove the following:

▶ **Theorem 7.** *There is an isometric embedding of rate $\frac{1}{8}$, i.e., for every positive integer n, there is an isometric embedding $\varphi_n : \{0,1\}^n \to \{0,1\}^{8n}$ of the Hamming metric into the edit metric.*

Consequently, we obtain a randomized communication complexity lower bound of $(\frac{1}{8} - o_\varepsilon(1))n$ for the Exact-Edit problem. Determining the optimal leading constant is an interesting open problem; in contrast to the Exact-Hamming problem, we suspect the randomized communication complexity of the Exact-Edit problem might be $(1-\delta)n$ bits for some constant $\delta > 0$.

Moreover, Theorem 7 gives, in a black-box way, edit-metric codes of positive constant rate with relative distance approaching $\frac{1}{16}$. Explicit constructions with larger relative distance are known, but they all rely on tailored analyses that exploit the structure of the codewords. Finally, a combinatorial consequence of Theorem 7 is the realization of the $\frac{n}{8}$-dimensional Hamming cube in the n-dimensional edit cube.

To prove Theorem 7, we introduce objects called *misaligners*, which can be thought of as codes in the edit metric with robust distance guarantees, in two distinct ways. First, misaligner codewords contain *wildcard symbols* and, irrespective of how these wildcards are instantiated, a large pairwise distance is guaranteed between any two distinct codewords. The second guarantee is even stronger: not only is the distance between pairs of codewords large, but the distance also remains large even when comparing a codeword with a concatenation of multiple codewords. This protects against situations where the concatenation of the suffix and prefix of two codewords might closely resemble another codeword. Formally defining misaligners requires a bit of work, but below, when we speak of (α, m)-misaligners, we informally refer to misaligners that have codewords of length m and relative distance α. We elaborate more on the parameters of misaligners in Section 2.2, and it is formally defined in in the full version of the paper [6].

In addition to misaligners, we also explicitly formulate objects called *ε-locally self-matching strings*, which are strings in which every substring of length ℓ has non-vertical matches of size at most $\varepsilon\ell$, where, by non-vertical matches, we forbid matching an index to itself. These objects appear implicitly in the context of synchronization strings introduced by Haeupler and Shahrasbi [19]. Again, we elaborate more on this notion in Section 2.2, and it is formally defined in the full version of the paper [6].

One of our main technical contributions is showing how to utilize the construction of misaligners and locally self-matching strings to design high-rate isometric embeddings of the Hamming metric into the edit metric.

▶ **Theorem 8** (Informal statement of Theorem 6.1 in [6])**.** *Suppose for some $\varepsilon, \alpha > 0$ and some integer m, there exists an (α, m)-misaligner and a ε-locally self-matching string, then for every positive integer n there is an isometric embedding $\varphi_n : \{0,1\}^n \to \{0,1\}^{Rn}$ of the Hamming metric into the edit metric, where:*

$$R = \left\lceil \frac{1}{(1-\varepsilon)\alpha - \frac{1}{3m-1}} \right\rceil .$$

Even with limited computing resources, we were able to construct a $(0.1625, 320)$-misaligner and a 0.224-locally self-matching string. This yields the embedding in Theorem 7 (We refer the reader to Section 8 of the full version of the paper [6] for more details).

It is possible that with sufficient time and computing resources (i.e., by making m large and ε tiny), we can use Theorem 8 to obtain embeddings with rate approaching α, where α is the relative distance parameter of the misaligner. From the current computer search, we speculate that α is above 0.2. Thus, it is possible in the future that we have an isometric embedding where n bits are mapped only to $5n$ bits.

As remarked earlier, locally self-matching strings are closely related to synchronization strings [19]. Both these objects are constructed using the algorithmic Lovasz Local Lemma [31]. Prior work established that for all positive integers n there exists a ε-locally self-matching string w of length n over alphabet of size $O(1/\varepsilon^2)$ [10]. However, we could not use any of the existing analyses using the Lovasz Local Lemma in the synchronization strings literature due to the large hidden constants in those works. Thus, en route to proving Theorem 7, we also improve the analysis of constructing synchronization strings. To the best of our knowledge, prior to our work, the hidden constant in the alphabet size bound in these analyses was about $4900 \cdot e^2 \approx 3.6 \times 10^4$, and we have reduced it to a quantity that approaches $e^2 \approx 7.39$ as ε goes to 0.

▶ **Theorem 9.** *Let Σ be a finite set and $\varepsilon \in \left(0, \frac{1}{2}\right]$ such that the following holds:*

$$|\Sigma| \geq \frac{e^2}{\varepsilon^2} \cdot \left(1 + 4\sqrt[4]{\varepsilon}\right).$$

Then, for all positive integers n, there exists a ε-locally self-matching string w over Σ of length n.

As an immediate consequence of this improved analysis, we also obtain the following (proved in the full version of the paper [6]).

▶ **Corollary 10.** *There exists an infinite* 0.999606*-synchronization string over an alphabet of size four.*

The above result adds to the work of [10] where it was shown that there is some unspecified constant ε such that ε-synchronization strings exist over an alphabet of size four.

1.3 Structure of Isometric Embeddings and Impossibility Results

The embedding in Theorem 8 is an example of an *interleaved embedding*, where the output string is formed by interleaving the input bits with sequences of bit strings that do not depend on the input. A natural question is whether there exist isometric embeddings, potentially with better rates, that do not follow this interleaving framework. We answer this negatively, demonstrating that isometry necessitates an interleaved structure in the output strings.

▶ **Theorem 11** (Isometry implies Interleaving)**.** *Every isometric embedding $\varphi : \{0,1\}^n \to \{0,1\}^N$ of the Hamming metric into the edit metric must be an interleaved embedding.*

A precise formulation of Theorem 11 requires a formal definition of interleaved embeddings, which we provide in Section 10 of the full version of the paper [6]. Informally, we naturally extend the intuitive notion of interleaving input bits with fixed bit patterns by allowing two additional flexibilities. First, we allow the input bits to appear *out-of-order* – e.g., the second input bit might appear after the first input bit in the output string. We further allow some of

the input bits to appear *complemented* in the output string. Thus, an isometric embedding of the Hamming metric into the edit metric is completely determined by three components – the order in which input bits appear in the output, the subset of input bits that are flipped, and the sequence of fixed bit strings that are to be interleaved with the input bits.

This structural constraint on Hamming-to-edit isometric embeddings enables us to derive bounds on the achievable rate. In particular, we show that for binary strings, any isometric embedding must stretch the input strings by a factor greater than 2.133.

▶ **Theorem 12.** *There exists a positive integer n_0 such that every isometric embedding $\varphi : \{0,1\}^n \to \{0,1\}^N$ of the Hamming metric into the edit metric with $n \geq n_0$ must have rate at most $\frac{15}{32}$.*

We note that the above upper bound on rate has been improved by Bhattacharya (see Theorem 4.30 in [5]) to $\frac{3}{7} + o(1)$.

Next, we generalize Theorem 12 to larger alphabets by first extending the definition of interleaved embeddings to larger alphabets (See Section 10.1 of the full version of the paper [6]), then proving that every isometric embedding must be an interleaved embedding, and finally showing an analogous statement to Theorem 12.

▶ **Theorem 13.** *For every alphabet Σ, there exists an integer n_0 such that every isometric embedding $\varphi : \Sigma^n \to \Sigma^N$ of the Hamming metric into the edit metric with $n \geq n_0$ must have rate at most $\frac{1}{2} - \frac{1}{16|\Sigma|}$.*

1.4 Isometric Embeddings over Larger Alphabets and Surpassing the 1/2 Rate Barrier

Having established the rate limitations for isometric embeddings over the binary alphabet, we next explore if leveraging a larger alphabet Σ enables the construction of embeddings $\varphi : \Sigma^n \to \Sigma^N$ with potentially improved rates. As in the binary case, we can obtain high-rate embeddings over Σ by using misaligners for larger alphabets. In fact, it is not hard to see that misaligners designed for the binary alphabet also function as misaligners for larger alphabets, without any loss in parameters. Therefore, increasing the alphabet size can only improve the achievable rate.

We show that even without relying on the full machinery of misaligners, one can obtain isometric embeddings with rates arbitrarily close to $\frac{1}{3}$ by making the alphabet large enough.

▶ **Theorem 14.** *For any $\rho > 0$, there exists an alphabet Σ such that for every positive integer n, there is an isometric embedding $\varphi_n : \Sigma^n \to \Sigma^N$ of the Hamming metric into the edit metric with rate at least $\frac{1}{3} - \rho$.*

Theorem 13 tells us that as long as the input and output alphabets remain the same, the rate in Theorem 14 can never be improved to $\frac{1}{2}$ or anything better. But what if we allow the input and output alphabets to be *different*?

In this setting, our notion of rate needs to be refined. In particular, when using the larger output alphabet, the appropriate measure to consider is not the lengths of strings but the *number of bits* contained in them. Thus, for an embedding $\varphi : \Sigma_{\text{in}} \to \Sigma_{\text{out}}$, we define the rate as $\frac{n \log |\Sigma_{\text{in}}|}{N \log |\Sigma_{\text{out}}|}$. With this refined definition, we show that not only can the $\frac{1}{2}$ rate barrier be surpassed, but the rate can actually be made *arbitrarily close to 1*!

▶ **Theorem 15.** *For every $\rho > 0$, there exist alphabets Σ_{in}, Σ_{out} such that for all positive integers n, there is an isometric embedding $\varphi_n : \Sigma_{\text{in}}^n \to \Sigma_{\text{out}}^N$ of the Hamming metric into the edit metric with rate at least $1 - \rho$.*

We provide a proof of Theorem 15 in Section 12 of the full version of the paper [6]. Theorem 14 is proven in Section 9 of the same full version [6].

1.5 Related works

In this subsection, we survey three lines of related works.

Embedding Hamming Metric into Edit Metric

As mentioned earlier, embeddings of the Hamming metric into the edit metric have primarily been studied to establish hardness results for problems under the edit distance. The first known instance of this, to the best of our knowledge, is due to Rubinstein [35], who gave a near-isometric embedding for binary strings with rate $\Theta(1/\log n)$ and used it to prove the hardness of the approximate closest pair problem under edit distance. In a subsequent work, Cohen-Addad *et al.* [12] presented a fully isometric embedding (in contrast to Rubinstein's near-isometry), albeit at the cost of a larger output alphabet. Their construction achieved a rate of $\Theta(1/(\log n \log\log n))$. The folklore isometric embedding referenced in Section 1.1 explicitly appears in [2].

Embedding Edit metric into Hamming metric

Embeddings in the other direction, i.e., from the edit metric to the Hamming metric, are abundant in the literature. A key motivation for embedding the edit metric into the Hamming metric is computational: Hamming distance can be computed efficiently in linear time, whereas edit distance cannot be solved in sub-quadratic time unless the Strong Exponential Time Hypothesis is false [4]. This underscores the need for approximation algorithms that can run in subquadratic time. One potential approach is to develop an embedding with low distortion, where the embedding process itself can be performed in subquadratic time.

The seminal work by Ostrovsky and Rabani [34] shows an embedding of the edit metric into the Hamming metric with distortion $2^{O(\sqrt{\log n \log\log n})}$. The rate achieved is $1/\log n$, and the embedding can be computed in polynomial time. There has been extensive research aimed at establishing lower bounds for the distortion of any such embedding, with the best-known lower bound being $O(\log n)$ [26].

An alternative approach to potentially improve the results above is through randomized embeddings. In this approach, the embedding function takes both an input string and a random string, and we measure the distortion when two strings are mapped using the same random string sequence. Within this framework, Johwari [24] showed the existence of $O(\log n \log^* n)$-distortion. Subsequently, [9] were the first to eliminate the dependence of the dimension in the distortion, achieving a quadratic distortion, i.e., if the edit distance between the input strings is at most k, then the Hamming distance between the embedded strings is bounded by $O(k^2)$.

Synchronization strings

Synchronization strings were introduced by Haeupler and Shahrasbi [19] in the context of codes able to tolerate insertion and deletion errors. In our context, the original application of synchronization strings can be viewed as embedding the Hamming metric near isometrically into the edit metric by increasing the alphabet size by a constant factor. Throughout the years, these strings have found uses in many settings, including interactive coding [21], locally decodable insertion-deletion codes [17], and list decodable insertion-deletion codes [20]. For a survey of the many uses of synchronization strings and how they are constructed, the reader is referred to [18].

1.6 Organization of the Proceedings Version

Due to the page limit for the conference proceedings, this version focuses on the proof overview and defers all formal proofs to the full version of the paper [6]. In Section 2, we provide an overview of the proofs for our main results. The full version contains the formal proof of Theorem 7 in Section 8, Theorem 14 in Section 9, Theorem 11 in Section 10, Theorem 13 in Section 11, and Theorem 15 in Section 12 [6]. In addition, misaligners and locally-self-similar strings – the two key ingredients for our high-rate embedding, are formally introduced and analyzed in Sections 4 and 5 of the same full version [6].

2 Proof Overview

In Section 2.1, we provide a proof overview of Theorem 1, which is our main conceptual contribution. We do not provide a formal proof of Theorem 1 as it is subsumed by Theorem 8 and Theorem 7. In Section 2.2, we provide a proof overview of Theorem 8, which is one of our main technical contributions. Furthermore, in Section 2.3, we summarize the key ideas behind our proof of Theorem 11, while in Section 2.4, we provide a proof sketch of Theorem 13.

2.1 Constant Rate Embeddings via Synchronization Strings

Our first observation is that if we are allowed an unbounded output alphabet, then there is a straightforward isometric embedding that takes any input string and interleaves it with a string with all distinct symbols. This embedding is clearly isometric since any misalignment of these new symbols results in mismatches, increasing the edit cost beyond that of the simple identity alignment, which equals the Hamming distance between the input strings. Moreover, the length of the output strings produced by this embedding is only twice that of the input strings.

However, our goal is to have a binary output alphabet, not an unbounded one. So, as an intermediate step, let us limit ourselves to a large but constant-sized output alphabet. Now, if we want to mimic our earlier embedding, we would need to find a string over a *constant-sized* alphabet that *approximates* the string with all distinct symbols. It turns out that objects known as *synchronization strings* [19] have the exact property that we want. Synchronization strings have many equivalent definitions. The one that is useful for our discussion is the following – a string is said to be an ε-synchronization string if, for every substring, its relative self-edit distance[4] is at least $1 - \varepsilon$. For every value of ε, one can efficiently construct ε-synchronization strings over an alphabet of size $\Theta(1/\varepsilon^2)$ [10]. We now illustrate how these strings can be used to construct constant-rate isometric embeddings over a large but fixed-size output alphabet.

To embed a string of length n, consider a $2/3$-synchronization string of length $3n$ over an alphabet Σ that is disjoint from the input alphabet $\{0, 1\}$. The embedding process, applied to an input string of length n, involves inserting 3 symbols from the synchronization string after every input symbol. It is evident that both the rate and the output alphabet size remain constant.

We now argue that this gives an isometric embedding. Suppose it does not; then there must be a pair of strings such that the edit distance between their embeddings is strictly smaller than their Hamming distance. Consider an optimal edit alignment of the embedded

[4] The self-edit distance of a string is the cost of the cheapest alignment converting the string to itself without matching any character to itself.

strings. It can be shown that this alignment partitions the strings into intervals, where for each interval in the partition, either the entire interval is mapped using the identity mapping, i.e., "vertical", or it is self-aligned such that none of its characters are mapped to themselves, i.e., "nowhere-vertical". See Figure 1 for an illustration.

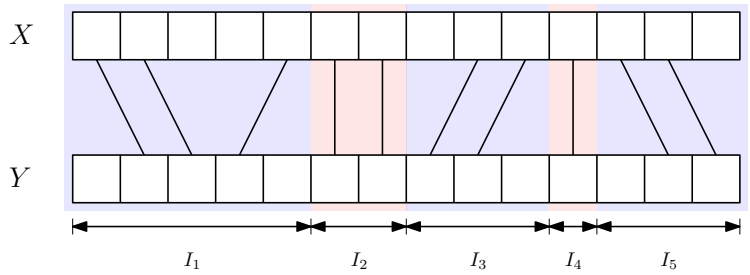


Figure 1 An optimal alignment converting the embedded string X to the embedded string Y. Note the alternating maximal nowhere-vertical and vertical intervals (highlighted blue and red, respectively) I_1, I_2, I_3, I_4 and I_5.

Now, if the edit distance is indeed strictly less than the Hamming distance, then there is at least one interval, say of length ℓ, in this partition where none of its characters are mapped to themselves, but the edit cost in that interval is strictly less than the Hamming cost, which is at most $\ell/4$. We then show that this implies the existence of a substring in the synchronization string with a relative self-edit distance less than $1/3$, which is a contradiction. This substring can be found by considering the aforementioned interval and deleting all the characters that come from the input strings. This results in a substring with length $\ell' := 3\ell/4$. One can also find a self-alignment of this substring by taking the original alignment, keeping it unchanged except for positions where some character in the substring was matched to some character in one of the input strings. In those positions, we simply apply a deletion. It is not hard to see that the cost of this alignment remains unchanged and is thus less than $\ell/4 = \ell'/3$, which contradicts the fact that the string used to pad is a $2/3$-synchronization string. In Figure 2, the red characters are part of some ε-synchronization string, and the green ones are part of some input string.

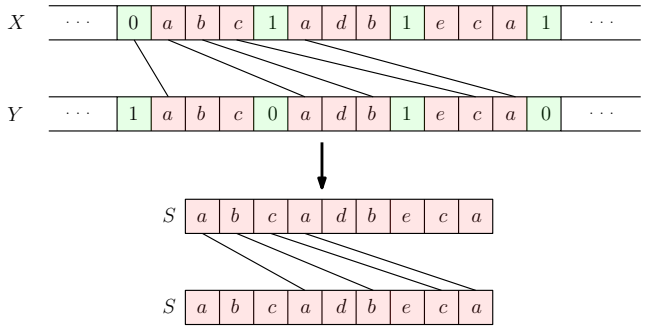


Figure 2 A nowhere-vertical alignment on some interval in the embedded strings X and Y implies a self-alignment of a substring S of the ε-synchronization string with the same cost.

One can then bring the output alphabet size down to two using the following alphabet reduction procedure – we replace each character of the synchronization string with its binary encoding (of length $\log|\Sigma|$) along with an opening and closing block of 0s and 1s with lengths equal to that of the binary encoding. The final resulting rate is still constant. However, the

large alphabet sizes required to construct ε-synchronization strings mean that the resulting rate is far from optimal. With the specific setting of $\varepsilon = 2/3$ and known bounds for alphabet sizes for synchronization strings, one gets a rate of about $1/153$ from this.

2.2 Optimizing the Rate: Enter Misaligners!

To improve the rate of our embedding, we move away from the framework of interleaving a synchronization string over a larger alphabet and then performing an alphabet reduction. Instead, we utilize a tailored construction that gives us a much larger rate.

The key ingredient in our construction is an object called a *misaligner*, which the reader should think of as a code in the edit metric with robust distance guarantees. A misaligner consists of a set of fixed-length codewords over the alphabet $\{0, 1, \star\}$, where $\star$ is a special wildcard symbol. Each codeword contains a wildcard symbol at every t^{th} position, where t is a parameter that controls the rate of the embedding. Thus, if one concatenates a sequence of codewords from the misaligner, one obtains a long binary string with wildcard symbols at every t^{th} position. To embed some input string, one simply substitutes these wildcard symbols with symbols from the input string. The rate of this embedding is $1/t$ by definition.

To fully describe the embedding, one also needs to specify the *order* in which codewords from the misaligner are concatenated. To do this, we make use of a variant of synchronization strings, which we call *locally self-matching strings*. Given $\varepsilon \in (0, 1)$, an ε-locally self-matching string is one where every substring of length ℓ has at most $\varepsilon\ell$ non-crossing symbol matches to itself if one is not allowed to match symbols vertically.[5] Given both these ingredients, we can now describe our embedding. To embed an input string, we first start with a sufficiently long locally self-matching string over some large alphabet. Next, we replace each symbol of this string with a codeword from the misaligner. Note that in order to do this, the number of codewords in the misaligner must be at least as large as the alphabet of the locally self-matching string. Finally, we substitute the wildcard symbols in this string with symbols of the input string to obtain the embedded string. Our goal is to show that by choosing t sufficiently large, the properties of the locally self-matching string and the misaligner guarantee that the resulting construction is an isometric embedding.

For concreteness, in the remainder of this section, let us assume that a 0.1-locally self-matching string s exists over some large alphabet Σ. We also assume that a misaligner $\mathcal{C}$ with $|\Sigma|$ many codewords exists. The other properties of this misaligner will be specified later. We will show how the properties of s and $\mathcal{C}$ will guarantee isometry.

Let us call a codeword with its wildcard symbols instantiated a *block*. Assume for the sake of contradiction that the embedding we just described is not an isometric embedding. As before, this implies the existence of an interval with an edit cost that is strictly lower than its Hamming cost. The first guarantee that the misaligner provides is that this interval must contain at least two full blocks. This is ensured by enforcing the codewords of the misaligner to have the following property – for any substring arising from the concatenation of multiple codewords and not containing two full blocks must have the edit distance equal to the Hamming distance, no matter how the wildcard symbols are instantiated. See Figure 3 for an illustration of this property.

For larger intervals (i.e., intervals containing at least two full blocks), we assume for simplicity that the interval consists solely of a sequence of full blocks with no partial blocks at the boundaries. The analysis then goes on to assess the cost of the optimal alignment by

[5] Alternatively, every substring has "nowhere-vertical" relative LCS at most ε; see Section 5 in the full version [6] for a formal definition.

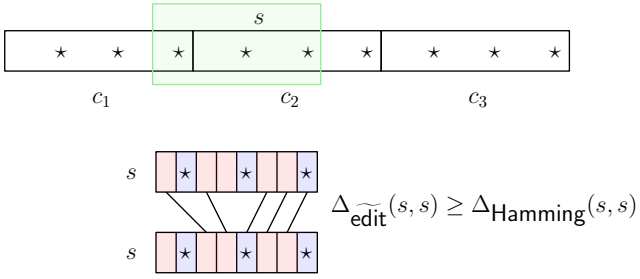


Figure 3 For short intervals, the misaligner guarantees isometry: any nowhere-vertical edit alignment must pay at least as much as the Hamming distance, no matter how the wildcards are instantiated.

decomposing it into contributions from individual blocks. Our objective is to demonstrate that, for most blocks, there is a substantial cost, which, for the purposes of this discussion, is 0.2 times the length of a block.

Our first observation concerns any pair of blocks within the specified interval that are instantiations of the same codeword but appear at different positions in the embedded strings. It can be shown (Obervation 7.1 in the full version [6]) that we can assume that in our optimal alignment, exactly one of the following cases occurs: either all of the characters of one of the blocks are matched "vertically" to all of the characters in the other block[6] (in which case, the first block is referred to as "bad"; see Figure 4), or none of the characters are matched "vertically". From the set of bad blocks and their matches, one can recover a sequence of nowhere-vertical matches in the original locally self-matching string s. Since s was a 0.1-locally self-matching string, one can thus conclude that the fraction of bad blocks is at most 0.1.

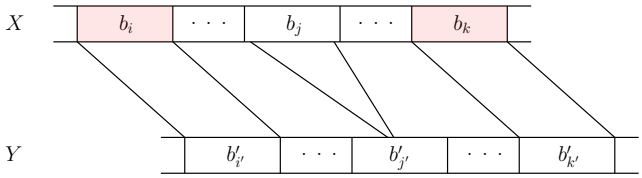


Figure 4 The blocks b_i and $b'_{i'}$ as well as b_j and $b'_{j'}$ and b_k and $b'_{k'}$ are instantiations of the same codeword and appear in different positions in X and Y. The bad blocks – those where all characters are matched vertically – are highlighted in red.

Next, let us analyze the remaining blocks. The optimal alignment transforms each of these blocks into a substring coming from the concatenation of multiple blocks. If the size of this substring exceeds 0.2 times the block size, then the cost will also exceed 0.2. For the remaining blocks, the misaligner must ensure that the edit cost of each block remains substantial. This can be accomplished by satisfying the following requirements:

Consider a block b and take *any* substring s of roughly the same size, composed of partial blocks. If none of these blocks is equal to b, we require that the relative edit cost between b and s is at least 0.2. If one of these blocks is equal to b, we require that the edit cost of alignments with no vertical edges between b and its counterpart in s is at least 0.2. See Figure 5 for an illustration.

[6] By "vertically", we mean the i^{th} character of the first block is matched against the i^{th} character of the second.

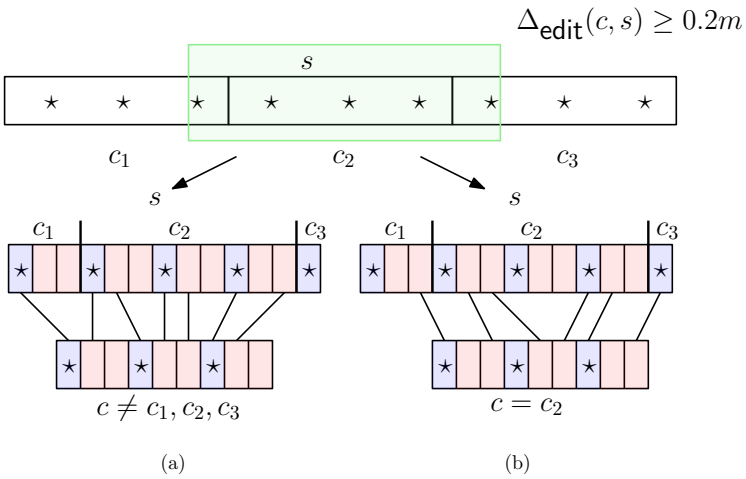


Figure 5 The alignment transforms every block c into a substring s arising from the concatenation of multiple blocks. If c and s are roughly the same size, the misaligner guarantees the relative edit distance between c and s is at least 0.2. Observe that in case (b), the alignment has no vertical edges between b and its counterpart in s.

Since bad blocks cost zero, the total cost is given by the sum of costs incurred by the non-bad blocks. Thus, by the properties of the misaligner and the locally self-matching string, the relative cost is lower bounded by $(1 - 0.1) \times 0.2 = 0.18$. Therefore, if we choose $t = 6$, we arrive at a contradiction, as the cost would then exceed the number of wildcards, which is an upper bound on the Hamming distance in the interval between any two strings. Consequently, we can achieve a $1/6$-rate isometric embedding. The case where the interval may contain partial blocks introduces additional requirements, which are detailed in Section 4 of the full version of the paper [6].

2.3 Unveiling the Interleaved Structure of Isometric Embeddings

In this section, we outline the key ideas behind our proof of Theorem 11, which states that every isometric embedding of the Hamming metric into the edit metric is necessarily an interleaved embedding. The formal definition of interleaved embeddings is given in Section 10 of the full version of the paper [6]. Here, we instead fix an arbitrary isometric embedding $\varphi : \{0,1\}^n \to \{0,1\}^N$ and make a series of observations that reveal certain constraints on φ, ultimately uncovering its interleaved nature.

We start by fixing an input string $x \in \{0,1\}^n$ and ask what happens to $\varphi(x)$ as we flip a single bit in x. Since φ is isometric, flipping a single bit in x must correspond to exactly one bit flip in $\varphi(x)$. This allows us to define a function $\eta_x : [n] \to [N]$, $\eta_x(i)$ is the unique index in $\varphi(x)$ that is flipped if the i^{th} bit of x is flipped.

Next, we show that the function η_x is *injective*, meaning that different bit flips in x affect different positions in the output $\varphi(x)$. This is not hard to see – if two different bit-flips in x affected the same position in $\varphi(x)$, then we would obtain a pair of strings at Hamming distance 2 whose images under φ have edit distance 0, contradicting the fact that φ is isometric.

A crucial step in the proof is showing that $\eta_x(i)$ does not depend on x. That is, once i is fixed, flipping the i^{th} bit in any input string always affects the same location in the output. This means the function η_x is the same for all inputs and can simply be written as η. This can be proven by fixing a pair of input strings x, y and an index i and inducting on the Hamming distance between x and y. This key step is presented in detail in Section 10 of the full version [6].

Finally, we show that $\varphi(x)[\eta(i)]$ depends only on $x[i]$, meaning that each input bit is mapped consistently to the same output position, either preserving or flipping its value. With this, the interleaved nature of φ is revealed: the output positions fall into two categories – those that directly correspond to input bits (determined by η) and the remaining positions that are independent of the input.

2.4 Upper Bounds on the Rate

In this section, we outline the proof of Theorem 13, which establishes that any isometric embedding of the Hamming metric into the edit metric must have a rate strictly bounded away from $\frac{1}{2}$. More precisely, we aim to show that for any alphabet Σ, there exists some $\varepsilon > 0$ such that for sufficiently large n, no isometric embedding $\varphi : \Sigma^n \to \Sigma^N$ can achieve a rate exceeding $\frac{1}{2} - \varepsilon$.

Our approach is by contradiction: we assume that for all n, there exists an isometric embedding φ with a rate exceeding the claimed bound. The goal then is to show that as n grows, φ cannot remain isometric. Specifically, we seek a pair of strings such that the edit distance between their embeddings is strictly smaller than their Hamming distance, contradicting isometry. Rather than constructing such a pair explicitly, we will argue its existence via the probabilistic method.

The key idea is to consider a collection $\mathcal{C}$ of carefully chosen triples $(x, y, \mathcal{A})$, where $x, y \in \Sigma^n$ and $\mathcal{A}$ is an edit distance alignment between $\varphi(x)$ and $\varphi(y)$. We then select a random triple from $\mathcal{C}$ and show that the *expected* cost of $\mathcal{A}$ with respect to the embeddings $\varphi(x)$ and $\varphi(y)$ is too low, ensuring the existence of a triple in $\mathcal{C}$ that violates the isometry of φ.

Description of $\mathcal{C}$

The alignments in $\mathcal{C}$ are simple "shift" alignments, where each symbol in one string is matched with a symbol in the other string shifted by a fixed amount. More formally, for a non-zero integer δ, we define $\mathcal{A}_\delta$ as the alignment where each symbol in the first string is paired with a symbol in the second string shifted by δ positions. We consider alignments of the form $\mathcal{A}_\delta$ where $|\delta|$ is bounded by a constant that depends on the rate we seek to rule out, in particular, ε, but is independent of n.

Since φ is isometric, by the generalization of Theorem 11 to larger alphabets (Theorem 10.9 in the full version [6]), it must be an interleaved embedding. This implies that some symbols in $\varphi(x)$ are "frozen" (independent of x), while the remaining symbols are "mutable" (determined by x). Using this observation, for each alignment $\mathcal{A}_\delta$, we construct a pair of strings $x_\delta, y_\delta \in \Sigma^n$ satisfying the following two properties.

- The Hamming distance between x_δ and y_δ is n.
- No mutable symbol in $\varphi(x_\delta)$ is substituted under $\mathcal{A}_\delta$; they are either matched or deleted.

Section 11 of the full version of the paper [6] details how to construct such pairs given $\mathcal{A}_\delta$. The final collection $\mathcal{C}$ consists of all triples $(x_\delta, y_\delta, \mathcal{A}_\delta)$ for δ in a constant-sized set of shifts.

Bounding the Expected Alignment Cost

We now pick a random $(x_\delta, y_\delta, \mathcal{A}_\delta)$ from $\mathcal{C}$ and analyze the expected cost of $\mathcal{A}_\delta$ with respect to $\varphi(x_\delta)$ and $\varphi(y_\delta)$. Since $\mathcal{A}_\delta$ is a shift alignment, its cost due to insertions and deletions is at most a constant. Moreover, by our choice of x_δ and y_δ, there are no substitutions involving

mutable symbols – only frozen symbols contribute to substitutions. Now if the rate of φ is too high, the number of frozen symbols must be small, which in turn means the expected number of substitutions is also small. We show that for sufficiently large n, the expected cost of $\mathcal{A}_\delta$ falls below n, leading to a contradiction. For the detailed calculations along with choices for the maximum value of δ, the reader is referred to Section 11 of the full version [6].

References

1 National Center for Biotechnology Information homepage. https://www.ncbi.nlm.nih.gov/. Accessed: 2024-11-04.

2 Amir Abboud, MohammadHossein Bateni, Vincent Cohen-Addad, Karthik C. S., and Saeed Seddighin. On complexity of 1-center in various metrics. In Nicole Megow and Adam D. Smith, editors, *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM 2023, September 11-13, 2023, Atlanta, Georgia, USA*, volume 275 of *LIPIcs*, pages 1:1–1:19. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2023. doi:10.4230/LIPIcs.APPROX/RANDOM.2023.1.

3 Alexandr Andoni and Robert Krauthgamer. The computational hardness of estimating edit distance. *SIAM J. Comput.*, 39(6):2398–2429, 2010. doi:10.1137/080716530.

4 Arturs Backurs and Piotr Indyk. Edit distance cannot be computed in strongly subquadratic time (unless SETH is false). *SIAM J. Comput.*, 47(3):1087–1097, 2018. doi:10.1137/15M1053128.

5 Sudatta Bhattacharya. String similarity measures: Metric embeddings and applications, 2025.

6 Sudatta Bhattacharya, Sanjana Dey, Elazar Goldenberg, Mursalin Habib, Bernhard Haeupler, Karthik C. S., and Michal Koucký. Constant rate isometric embeddings of hamming metric into edit metric, 2025. doi:10.48550/arXiv.2504.03605.

7 Amit Chakrabarti, Graham Cormode, and Andrew McGregor. A near-optimal algorithm for estimating the entropy of a stream. *ACM Trans. Algorithms*, 6(3):51:1–51:21, 2010. doi:10.1145/1798596.1798604.

8 Amit Chakrabarti and Oded Regev. An optimal lower bound on the communication complexity of gap-hamming-distance. *SIAM J. Comput.*, 41(5):1299–1317, 2012. doi:10.1137/120861072.

9 Diptarka Chakraborty, Elazar Goldenberg, and Michal Koucký. Streaming algorithms for embedding and computing edit distance in the low distance regime. In Daniel Wichs and Yishay Mansour, editors, *Proceedings of the 48th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2016, Cambridge, MA, USA, June 18-21, 2016*, pages 712–725. ACM, 2016. doi:10.1145/2897518.2897577.

10 Kuan Cheng, Bernhard Haeupler, Xin Li, Amirbehshad Shahrasbi, and Ke Wu. Synchronization strings: Highly efficient deterministic constructions over small alphabets. In *Proceedings of the ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 2185–2204, 2019. doi:10.1137/1.9781611975482.132.

11 Benny Chor and Oded Goldreich. Unbiased bits from sources of weak randomness and probabilistic communication complexity. *SIAM J. Comput.*, 17(2):230–261, 1988. doi:10.1137/0217015.

12 Vincent Cohen-Addad, Laurent Feuilloley, and Tatiana Starikovskaya. Lower bounds for text indexing with mismatches and differences. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1146–1164. SIAM, 2019. doi:10.1137/1.9781611975482.70.

13 Vincent Cohen-Addad, Karthik C. S., and Euiwoong Lee. Johnson coverage hypothesis: Inapproximability of k-means and k-median in lp-metrics. In Joseph (Seffi) Naor and Niv Buchbinder, editors, *Proceedings of the 2022 ACM-SIAM Symposium on Discrete Algorithms, SODA 2022, Virtual Conference / Alexandria, VA, USA, January 9 - 12, 2022*, pages 1493–1530. SIAM, 2022. doi:10.1137/1.9781611977073.63.

14 Henry L. Fleischmann, Surya Teja Gavva, and Karthik C. S. On approximability of steiner tree in lp-metrics. In David P. Woodruff, editor, *Proceedings of the 2024 ACM-SIAM Symposium on Discrete Algorithms, SODA 2024, Alexandria, VA, USA, January 7-10, 2024*, pages 1669–1703. SIAM, 2024. doi:10.1137/1.9781611977912.67.

15 Xinbo Gao, Bing Xiao, Dacheng Tao, and Xuelong Li. A survey of graph edit distance. *Pattern Analysis and applications*, 13:113–129, 2010. doi:10.1007/S10044-008-0141-Y.

16 Uri Hadar, Jingbo Liu, Yury Polyanskiy, and Ofer Shayevitz. Communication complexity of estimating correlations. In *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing*, pages 792–803, 2019. doi:10.1145/3313276.3316332.

17 Bernhard Haeupler and Amirbehshad Shahrasbi. Synchronization strings: Explicit constructions, local decoding, and applications. In *Proceedings of the ACM Symposium on Theory of Computing (STOC)*, pages 841–854, 2018. doi:10.1145/3188745.3188940.

18 Bernhard Haeupler and Amirbehshad Shahrasbi. Synchronization strings and codes for insertions and deletions - a survey. *CoRR*, abs/2101.00711, 2021. arXiv:2101.00711.

19 Bernhard Haeupler and Amirbehshad Shahrasbi. Synchronization strings: Codes for insertions and deletions approaching the singleton bound. *J. ACM*, 68(5):36:1–36:39, 2021. doi:10.1145/3468265.

20 Bernhard Haeupler, Amirbehshad Shahrasbi, and Madhu Sudan. Synchronization strings: List decoding for insertions and deletions. In *Proceedings of the International Conference on Automata, Languages, and Programming (ICALP)*, 2018.

21 Bernhard Haeupler, Amirbehshad Shahrasbi, and Ellen Vitercik. Synchronization strings: Channel simulations and interactive coding for insertions and deletions. In *Proceedings of the International Conference on Automata, Languages, and Programming (ICALP)*, pages 75:1–75:14, 2018. doi:10.4230/LIPIcs.ICALP.2018.75.

22 Piotr Indyk and David P. Woodruff. Tight lower bounds for the distinct elements problem. In *44th Symposium on Foundations of Computer Science, FOCS 2003, Cambridge, MA, USA, October 11-14, 2003, Proceedings*, pages 283–288. IEEE Computer Society, 2003. doi:10.1109/SFCS.2003.1238202.

23 Piotr Indyk and David P. Woodruff. Optimal approximations of the frequency moments of data streams. In Harold N. Gabow and Ronald Fagin, editors, *Proceedings of the 37th Annual ACM Symposium on Theory of Computing, Baltimore, MD, USA, May 22-24, 2005*, pages 202–208. ACM, 2005. doi:10.1145/1060590.1060621.

24 Hossein Jowhari. Efficient communication protocols for deciding edit distance. In Leah Epstein and Paolo Ferragina, editors, *Algorithms - ESA 2012 - 20th Annual European Symposium, Ljubljana, Slovenia, September 10-12, 2012. Proceedings*, volume 7501 of *Lecture Notes in Computer Science*, pages 648–658. Springer, 2012. doi:10.1007/978-3-642-33090-2_56.

25 Karthik C. S. and Pasin Manurangsi. On closest pair in euclidean metric: Monochromatic is as hard as bichromatic. *Combinatorica*, 40(4):539–573, 2020. doi:10.1007/S00493-019-4113-1.

26 Robert Krauthgamer and Yuval Rabani. Improved lower bounds for embeddings into L_1. *SIAM J. Comput.*, 38(6):2487–2498, 2009. doi:10.1137/060660126.

27 Eyal Kushilevitz and Noam Nisan. *Communication Complexity*. Cambridge University Press, New York, NY, USA, 1997.

28 Ming Li, Bin Ma, and Lusheng Wang. On the closest string and substring problems. *Journal of the ACM*, 49, March 2002. doi:10.1145/506147.506150.

29 Anders Liljas, Lars Liljas, Goran Lindblom, Poul Nissen, Morten Kjeldgaard, and Miriam-rose Ash. *Textbook of structural biology*, volume 8. World Scientific, 2016.

30 Andres Marzal and Enrique Vidal. Computation of normalized edit distance and applications. *IEEE transactions on pattern analysis and machine intelligence*, 15(9):926–932, 1993. doi:10.1109/34.232078.

31 Robin A Moser and Gábor Tardos. A constructive proof of the general lovász local lemma. *Journal of the ACM (JACM)*, 57(2):1–15, 2010. doi:10.1145/1667053.1667060.

32 Gonzalo Navarro. A guided tour to approximate string matching. *ACM computing surveys (CSUR)*, 33(1):31–88, 2001. doi:10.1145/375360.375365.

33 François Nicolas and Eric Rivals. Hardness results for the center and median string problems under the weighted and unweighted edit distances. *Journal of Discrete Algorithms*, 3(2):390–415, 2005. Combinatorial Pattern Matching (CPM) Special Issue. doi:10.1016/j.jda.2004.08.015.

34 Rafail Ostrovsky and Yuval Rabani. Low distortion embeddings for edit distance. *J. ACM*, 54(5):23, 2007. doi:10.1145/1284320.1284322.

35 Aviad Rubinstein. Hardness of approximate nearest neighbor search. In Ilias Diakonikolas, David Kempe, and Monika Henzinger, editors, *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2018, Los Angeles, CA, USA, June 25-29, 2018*, pages 1260–1268. ACM, 2018. doi:10.1145/3188745.3188916.

36 Leonard J Schulman and David Zuckerman. Asymptotically good codes correcting insertions, deletions, and transpositions. *IEEE transactions on information theory*, 45(7):2552–2557, 1999. doi:10.1109/18.796406.

37 Alexander A Sherstov. The communication complexity of gap hamming distance. *Theory of Computing*, 8(1):197–208, 2012. doi:10.4086/TOC.2012.V008A008.

38 Thomas Vidick. A concentration inequality for the overlap of a vector on a large set. *Chicago Journal of Theoretical Computer Science*, 1:1–12, 2012.

39 Robert A. Wagner and Michael J. Fischer. The string-to-string correction problem. *J. ACM*, 21(1):168–173, January 1974. doi:10.1145/321796.321811.