THEORY OF COMPUTING, Volume 2 (2006), pp. 207–224
http://theoryofcomputing.org

Embedding the Ulam metric into ℓ_1

Moses Charikar* Robert Krauthgamer

Received: December 8, 2005; revised: September 18, 2006; published: September 29, 2006.

Abstract: Edit distance is a fundamental measure of distance between strings, the extensive study of which has recently focused on computational problems such as nearest neighbor search, sketching and fast approximation. A very powerful paradigm is to map the metric space induced by the edit distance into a normed space (e. g., ℓ_1) with small distortion, and then use the rich algorithmic toolkit known for normed spaces. Although the minimum distortion required to embed edit distance into ℓ_1 has received a lot of attention lately, there is a large gap between known upper and lower bounds. We make progress on this question by considering large, well-structured submetrics of the edit distance metric space.

Our main technical result is that the Ulam metric, namely, the edit distance on permutations of length at most n, embeds into ℓ_1 with distortion $O(\log n)$. This immediately leads to sketching algorithms with constant size sketches, and to efficient approximate nearest neighbor search algorithms, with approximation factor $O(\log n)$. The embedding and its algorithmic consequences present a big improvement over those previously known for the Ulam metric, and they are significantly better than the state of the art for edit distance in general. Further, we extend these results for the Ulam metric to edit distance on strings that are (locally) non-repetitive, i. e., strings where (close by) substrings are distinct.

ACM Classification: F.2.2, G.2.1, G.3

AMS Classification: 68P05, 68W20, 68W25

Key words and phrases: edit distance, metric embedding, Ulam metric, low distortion, sketching, permutation edit distance

*Supported by NSF ITR grant CCR-0205594, DOE Early Career Principal Investigator award DE-FG02-02ER25540, NSF CAREER award CCR-0237113, MSPA-MCS award 0528414, and an Alfred P. Sloan Fellowship

 DOI: 10.4086/toc.2006.v002a011

1 Introduction

The *edit distance* (aka *Levenshtein distance*) between two strings is the least number of character insertions, deletions or substitutions required to transform one string to the other. The edit distance arises naturally in several application areas, often involving large amounts of data, ranging from a moderate number of extremely long strings (as in computational biology) to a large number of moderately long strings (as in text processing and web search). Therefore, efficient algorithms for edit distance, even with modest approximation guarantees, are highly desirable.

The edit distance is a fundamental measure of (dis)similarity, because it is a very simple model that exhibits nontrivial alignment (i. e., a single elementary modification may cause numerous characters to change their position in the string). Popular metrics such as the Hamming distance do not adequately capture this phenomenon, and thus for data analysis purposes, edit distance often offers a much better model (up to minor variations such as weighting).

Let $\mathrm{ED}(x,y)$ denote the edit distance between strings x and y. Fixing an alphabet Σ, the edit distance function $\mathrm{ED}(\cdot,\cdot)$ defines a metric space, called the *edit distance metric*, whose point set contains all the strings over Σ. Every collection X of strings over this alphabet (e. g., $X = \{0,1\}^n$) can therefore be associated with the submetric (X,ED). A significant obstacle in dealing with the edit distance metric is that it lacks many of the useful properties of normed spaces. We restrict our attention to collections X of strings that have limited repetitions. While these collections of strings are rather large and give rise to submetrics (X,ED) of a rich structure, we will be able to obtain results that are significantly better than those known for the general case $X = \Sigma^n$ (or even $\{0,1\}^n$). Our main focus is the Ulam metric, which we define next.

The Ulam Metric Following [7, 5], a string (over the alphabet Σ) is called a *permutation* if its characters are all distinct. (This notion, sometimes called a variation, extends the usual notion of permutation to cases where $|\Sigma| > n$, but this slightly more general setting is more convenient for our purposes and potentially more useful in applications.) Throughout this paper, the *Ulam metric* of dimension n (called permutation edit distance in [7]) is the metric space $(\mathcal{P}_n,\mathrm{ED})$, where $\mathcal{P}_n$ contains all permutations of length n over Σ.

Remark: The standard definition of the Ulam metric for permutations (see e. g. [1]) uses the distance function $\mathrm{UL}(x,y)$, defined as the least number of character moves needed to transform x into y. This distance function is obviously limited to the case $n = |\Sigma|$ (i. e. to the usual notion of permutations), while it is easily seen to be nearly equivalent to the edit distance, namely $\mathrm{UL}(x,y) \le \mathrm{ED}(x,y) \le 2\,\mathrm{UL}(x,y)$. Thus, our non-standard definition of the Ulam metric to be $(\mathcal{P}_n,\mathrm{ED})$ is merely a slight abuse of terminology to gain additional generality.

Embeddings An *embedding* of a metric space (X,d_X) into a target metric space (Y,d_Y) is a mapping $f : X \to Y$. The *distortion* of the embedding f is defined as the smallest $K \ge 1$ for which there is (a scaling factor) $s > 0$ such that

$$d_X(x_1,x_2) \le s \cdot d_Y(f(x_1), f(x_2)) \le K \cdot d_X(x_1,x_2) \qquad \text{for all } x_1,x_2 \in X \ .$$

Low-distortion embeddings are a very powerful paradigm for reducing a host of problems (such as Nearest Neighbor Search, see Section 3) from a given metric space to a more structured or computationally easier metric space, at the cost of a small loss in the approximation guarantee. It is easy to see that the Ulam metric contains the $\lfloor n/2 \rfloor$-dimensional hypercube as a submetric (up to scaling),[1] and hence, by Enflo's Theorem [9], embedding the Ulam metric into ℓ_2 requires distortion $\Omega(\sqrt{n})$. But an ℓ_1 embedding is usually sufficient for algorithmic applications such as sketching algorithms and Nearest Neighbor Search (see Section 3.3 for definitions), which raises the question of embedding the Ulam metric, and more generally the edit distance metric, into ℓ_1.

There is a very large gap between the known upper and lower bounds on the distortion required to embed the edit distance metric (Σ^n, ED) into ℓ_1. Ostrovsky and Rabani [20] showed an upper bound of $2^{O(\sqrt{\log n \log\log n})}$ for embedding edit distance into ℓ_1. Very recently, Khot and Naor [16] obtained a lower bound of $\Omega(\sqrt{\log n / \log\log n})$, and this was further improved to $\Omega(\log n)$ by Krauthgamer and Rabani [17]. For the Ulam metric, the same upper bound $2^{O(\sqrt{\log n \log\log n})}$ clearly holds and is still the state of the art. On the other hand, Cormode [5, page 60] shows a lower bound of $4/3$, using a 5-point metric that is isomorphic to the $K_{2,3}$ graph (up to scaling). Our embedding results make progress towards resolving these intriguing questions by proving an exponentially smaller upper bound for the Ulam metric, and extending it to several related edit distance submetrics.

1.1 Results

Our main result, appearing in Section 2, is that the n-dimensional Ulam metric embeds into ℓ_1 with distortion $O(\log n)$.[2] The previously known upper bound is $2^{O(\sqrt{\log n \log\log n})}$, using the embedding of [20] for edit distance in general. Our embedding is surprisingly simple and easy to describe. In fact, this is the complete description: we have a coordinate for every pair of symbols $a, b \in \Sigma$ and the value of this coordinate in the embedding of a string is simply the inverse of the distance between a and b in the string (or 0 if one of a, b does not occur).

Techniques Our methodology is inspired by the work of Cormode, Muthukrishnan and Sahinalp [7], who designed a mapping from the Ulam metric to Set-Intersection.[3] However, their mapping is not an embedding into ℓ_1 (in fact, Set-Intersection is not even a metric space) and it does not yield a sketching algorithm for the Ulam metric. The main difficulty in the analysis of our embedding is to prove that it is not too contractive. To this end, we build on the framework of [7], in which a common subsequence for two permutations is constructed recursively by partitioning each permutation into two subsequences. The main novelty in our analysis is that we replace the deterministic recursive partitioning of [7] with a carefully-crafted stochastic partitioning, resulting in a suitable averaging over all symbol pairs.

[1] Consider e. g. the permutations P for which $\{P(2i-1), P(2i)\} = \{2i-1, 2i\}$ for all $i = 1, \ldots, \lfloor n/2 \rfloor$.

[2] This metric space contains more than 2^n points, and thus our distortion bound beats by far the one that follows from Bourgain's embedding theorem [4] for general finite metrics. Note that in the nearest neighbor search setting, one needs to embed not only S into ℓ_1, but also the (yet unknown) query point.

[3] Namely, every permutation is mapped to a subset of some fixed ground set U, such that the edit distance between two permutations is approximately the size of the intersection between the two respective subsets.

Applications The Ulam metric is interesting in its own right, e. g. when considering the Ulam metric as a measure of (dis)similarity between rankings for say aggregation purposes. But it is not clear a priori that results on the Ulam metric would lend themselves to broader classes of strings; bit strings, for instance, cannot have distinct characters unless $n \leq 2$. Nevertheless, we show in Section 3 several applications of the above embedding result to edit distance on more general strings.

Let us mention one generalization of the above embedding here, leaving further discussion to Section 3. Following [2], we say that a string is *t-non-repetitive*, if all its t-substrings are distinct. For example, a random bit string of length n is, with high probability, $2\log n$-non-repetitive. We can easily extend the above embedding to t-non-repetitive strings with a $2t$ factor loss in the distortion. For instance, this embedding is applicable, with high probability, for two random, but correlated, bit strings (e. g., x is chosen uniformly at random and y is derived from it by deleting certain positions). Such scenarios often arise in computational biology contexts due to background distributions.

All our embeddings (e. g. the one specified above) are efficiently computable, and are thus readily applicable to computational problems. For instance, they immediately yield sketching algorithms and Nearest Neighbor Search schemes, as stated in Section 3. The algorithms that we obtain for restricted families of strings all have significantly better approximation guarantees than the state of the art for edit distance in general [14, 2, 20]. For one thing, restriction to strings with limited repetitions may be reasonable in many specific scenarios, and may serve as a rigorous starting point for domain-specific heuristics.

In addition, our results make partial progress on the general case; they identify algorithmic tools that are provably useful (in certain cases), and they pinpoint some difficult aspects (of the general case). The recent embedding result of Ostrovsky and Rabani [20] relies on a recursive construction. Our results on the Ulam metric and its extensions suggest that it may be possible to achieve a polylogarithmic distortion for embedding general edit distance. However, achieving this bound may require going beyond recursive constructions and using a "magical" direct embedding along the lines of the one we construct.

1.2 Related work

Cormode, Muthukrishnan and Sahinalp [7] were the first to suggest embeddings of various distance functions on permutations. For reversal distance and transposition distance they designed embeddings into Hamming space with constant distortion. However, as mentioned earlier, for edit distance (on permutations) they designed a mapping into Set-Intersection, which cannot be embedded into ℓ_1 or yield a sketching algorithm. They also use this mapping into Set-Intersection to obtain a fast approximate string matching for permutations (under edit distance).

Cormode and Muthukrishnan [6] show that a variant of edit distance called the *block edit distance*, where a block of characters can be moved in a single edit operation, embeds into ℓ_1 with distortion $O(\log n \log^* n)$. See also [8, 19] for embeddings of similar distance function on strings.

Batu et al. [3] developed a sub-linear time algorithm that runs in $O(n^{\max(\alpha/2, 2\alpha-1)})$ time and solves the $O(n^\alpha)$ vs. $\Omega(n)$ edit distance gap problem. Their algorithm can be cast as a sketching algorithm, although it would use a sketch whose size is far more than constant.

1.3 Notation

As usual, define $[k] = \{1,\dots,k\}$. A *string* is a sequence of characters (i. e., symbols) taken from an alphabet Σ. The jth character in a string s is denoted by $s[j]$. We write $a \in s$ to denote that a character $a \in \Sigma$ appears in the string s, and $a \notin s$ otherwise. A *t-substring* (aka *t-gram*) of a string s is a string consisting of t consecutive characters in s, e. g. $s[i], s[i+1], \dots, s[i+t-1]$. In contrast, a *subsequence* of s need not be contiguous in s.

A *permutation* is string P whose characters are all distinct. For permutations, it will sometimes be more convenient to work with P^{-1}, i. e., $P^{-1}(a)$ is the position at which $a \in \Sigma$ appears in the string P (if at all).

Longest common subsequence and edit distance For two strings x,y, let $\mathrm{LCS}(x,y)$ be the length of the longest common subsequence of x and y (i. e., the maximum length of a string z that is both a subsequence of x and a subsequence of y). It is well-known and easy to verify that for every two strings (and in particular permutations) x,y of length n, we have $n - \mathrm{LCS}(x,y) \le \mathrm{ED}(x,y) \le 2(n - \mathrm{LCS}(x,y))$.

2 Embedding the Ulam metric

In this section we present a low-distortion embedding of the Ulam metric (i. e., the edit distance metric on permutations) into ℓ_1.

Theorem 2.1. *For every n, the Ulam metric of dimension n can be embedded into $\ell_1^{O(|\Sigma|^2)}$ with distortion $O(\log n)$.*

Fix an integer n; we may assume that n is a power of 2, e. g., by padding all strings using additional characters. Let $m = |\Sigma|$, and assume without loss of generality that $\Sigma = \{1,\dots,m\}$.

Define an embedding $f : \mathcal{P}_n \to \ell_1^{\binom{m}{2}}$ as follows. First, associate every coordinate of the target space with a distinct pair $\{a,b\}$ where $a,b \in \Sigma$ and $a \neq b$. Now for every permutation $P \in \mathcal{P}_n$, the coordinates of $f(P)$ are given by

$$f(P)_{\{a,b\}} := \begin{cases} 1/(P^{-1}(b) - P^{-1}(a)) & \text{if } a,b \in P,\ a<b; \\ 0 & \text{otherwise (i. e., } a \notin P \text{ or } b \notin P). \end{cases} \tag{2.1}$$

The proof of Theorem 2.1 is completed below in Lemma 2.2 and Lemma 2.3, which analyze the expansion and contraction of this embedding f, respectively.

Lemma 2.2 (Expansion). *Let P and Q be permutations of length n. Then*

$$\|f(P) - f(Q)\|_1 \le O(\log n) \cdot \mathrm{ED}(P,Q) \ .$$

Proof. Extend f to permutations of length at most n by using the same definition (2.1). Observe now that it suffices to prove the claimed inequality $\|f(P) - f(Q)\|_1 \le O(\log n) \cdot \mathrm{ED}(P,Q)$ for the case where $\mathrm{ED}(P,Q) = 1$, the length of P is n and the length of Q is $n-1$. Indeed, the general case then follows by the triangle inequality in ℓ_1. (In the process, we simulate a character substitution by a deletion followed by an insertion, which increases the number of character operations at most by a factor of 2.)

Suppose then that Q is obtained from P by deleting some character $P[s]$. Thus, we have (i) $Q[i] = P[i]$ for all $i < s$ and (ii) $Q[i] = P[i+1]$ for all $i \geq s$. Clearly,

$$\|f(P) - f(Q)\|_1 = \sum_{a,b \in \Sigma} \left| f(P)_{\{a,b\}} - f(Q)_{\{a,b\}} \right| \quad .$$

It suffices to consider only the terms in which $a \in P$ and $b \in P$, as every other term is 0. So suppose $a = P[i]$ and $b = P[j]$ for $i < j$. In the case where $i = s$, clearly $f(Q)_{\{a,b\}} = 0$; thus, the total contribution of this case is $\sum_{j=s+1}^{n} \frac{1}{j-s} \leq H(n)$ where $H(k) = \sum_{z=1}^{k} \frac{1}{z}$ is the kth harmonic number. The case where $j = s$ is similar, with $f(Q)_{\{a,b\}} = 0$ and thus total contribution $\sum_{i=1}^{s-1} (\frac{1}{s-i}) \leq H(n)$. The case where both i and j are smaller than s, as well as the case where both i and j are larger than s, contribtes zero since $f(P)_{\{a,b\}} = f(Q)_{\{a,b\}} = \frac{1}{j-i}$. The last case where $i < s < j$ has total contribution at most

$$\sum_{i=1}^{s-1} \sum_{j=s+1}^{\infty} \left| \frac{1}{j-i} - \frac{1}{j-i-1} \right| \quad ;$$

for each i, the summation over j is a telescopic sum bounded above by $\frac{1}{s-i}$, implying that the total contribution of this case is at most $H(n)$. Hence, $\|f(P) - f(Q)\|_1 \leq 3H(n) \leq 3(1 + \ln n) = O(\log n)$. $\square$

Our proof method of the next lemma, which bounds the contraction of f, is inspired by the work of Cormode, Muthukrishnan and Sahinalp [7]. At a high level, we recursively construct a common subsequence by first partitioning the alphabet, thereby partitioning each string into two subsequences, and then merging the two common subsequences obtained by recursion. Our analysis is more involved than that of [7]. In particular, we employ a carefully-crafted stochastic partitioning that "smooths" the effect of any single pair of characters.

Lemma 2.3 (Contraction). *Let P and Q be permutations of length n, and assume n is a power of 2. Then $\|f(P) - f(Q)\|_1 \geq \frac{1}{8} \mathrm{ED}(P, Q)$.*

Before proving this lemma, we introduce some definitions and a technical proposition. Let P be a permutation of length k. Denote by $\mathrm{LIS}(P)$ the length of a longest increasing subsequence of P. Define a *breakpoint* in P to be a position $i \in [k-1]$ where $P[i] > P[i+1]$, and denote by $\mathrm{b}(P)$ the number of breakpoints in P. Two subsequences of P are called a *partition* of P if each character of P appears in exactly one of the two subsequences. A *block* is a pair of positions $\{2i-1, 2i\}$ where $i \in [\lfloor k/2 \rfloor]$. A partition of P into two subsequences P_0, P_1 is called *block-balanced* if, at every block $\{2i-1, 2i\}$, exactly one of the two characters belongs to P_0 (hence also exactly one of them belongs to P_1). Note that if the length of P is even and a partition of P is block-balanced, then the two corresponding subsequences have equal length.

Proposition 2.4. *Let P be a permutation of length k, and suppose k is even. Then for every block-balanced partition of P into subsequences P_0 and, P_1,*

$$\mathrm{LIS}(P) \geq \mathrm{LIS}(P_0) + \mathrm{LIS}(P_1) - 2\,\mathrm{b}(P) \quad .$$

Proof. We will actually prove that

$$\mathrm{LIS}(P) \geq 2\,\mathrm{LIS}(P_0) - 2\,\mathrm{b}(P) \ . \tag{2.2}$$

A symmetric argument for P_1 will imply similarly that $\mathrm{LIS}(P) \geq 2\,\mathrm{LIS}(P_1) - 2\,\mathrm{b}(P)$, and the lemma will follow by averaging the last two inequalities.

Let us then prove (2.2). Fix a longest increasing subsequence of P_0, and, with a slight abuse of notation, let $\mathrm{LIS}(P_0)$ denote both this subsequence and its length. We construct an increasing subsequence of P, by augmenting $\mathrm{LIS}(P_0)$ with certain characters from P_1. For each position $j \in [k]$ such that $P[j]$ is in P_0, let $j' \in \{j-1, j+1\}$ be the position such that $\{j, j'\}$ forms a block. Notice that $P[j']$ is in P_1, and call it a *candidate* if the corresponding $P[j]$ is in $\mathrm{LIS}(P_0)$. The number of candidates is thus exactly $\mathrm{LIS}(P_0)$. In particular, if augmenting $\mathrm{LIS}(P_0)$ with all the candidates forms an increasing subsequence of P, then the length of this increasing subsequence would be $2\,\mathrm{LIS}(P_0)$, and it would prove (2.2). We show next that $\mathrm{LIS}(P_0)$ can be always be augmented with $\mathrm{LIS}(P_0) - 2\,\mathrm{b}(P)$ candidates.

Consider two consecutive characters of $\mathrm{LIS}(P_0)$, say $P[j_1]$ and $P[j_2]$ with $j_1 < j_2$. (Essentially the same argument works in the two extremal cases where j_2 is the first character of $\mathrm{LIS}(P_0)$ and where j_1 is the last character of $\mathrm{LIS}(P_0)$.) Let t be the number of candidates among $P[j_1+1], \ldots, P[j_2-1]$. We claim that $t \leq 2$; indeed, only j_1+1 and j_2-1 can possibly be candidates, because if $P[j']$ is a candidate for $j' \in \{j_1+1, \ldots, j_2-1\}$ then for the corresponding $P[j]$ in $\mathrm{LIS}(P_0)$ we have $j \in \{j'-1, j'+1\} \subseteq \{j_1, \ldots, j_2\}$ and thus $j \in \{j_1, j_2\}$. If the t candidates are themselves in increasing order and they are all between $P[j_1]$ and $P[j_2]$, then augment $\mathrm{LIS}(P_0)$ with these t candidates; clearly, the result is still an increasing subsequence of P. Otherwise, P must contain some breakpoint $\hat{j} \in \{j_1, \ldots, j_2-1\}$, and this breakpoint can be blamed for not augmenting $\mathrm{LIS}(P_0)$ with these $t \leq 2$ candidates. Applying this augmentation for every two consecutive characters $P[j_1]$ and $P[j_2]$ of $\mathrm{LIS}(P_0)$, we see that every breakpoint is blamed only for candidates that are in the same interval $\{j_1, \ldots, j_2-1\}$ as the breakpoint, and thus every breakpoint is blamed for a total of at most 2 candidates. It follows that we have augmented $\mathrm{LIS}(P_0)$ with at least $\mathrm{LIS}(P_0) - 2\,\mathrm{b}(P)$ candidates. It is easy to see that this results in an increasing subsequence of P (because augmenting at one interval does not prevent augmenting at another interval) and this proves (2.2). ☐

Proof of Lemma 2.3. Start with the two length n permutations P and Q, and recall that we used padding to make n be a power of 2. We may assume that P and Q contain exactly the same characters, because every character $a \in \Sigma$ that appears in exactly one of the two strings contributes 1 to $\mathrm{ED}(P,Q)$ and at least 2 (actually $\Omega(\log n)$) to its "own" coordinates (all $\{a,b\}$ where $b \in \Sigma \setminus a$) in the difference vector $f(P) - f(Q)$. We can further assume (by renaming characters in Σ) that Q is the identity permutation, i. e., $Q[i] = i$. Hence,

$$\mathrm{ED}(P,Q) \leq 2(n - \mathrm{LCS}(P,Q)) = 2(n - \mathrm{LIS}(P)) \ .$$

Pick a random block-balanced partition of P into subsequences P_0 and P_1, i. e., independently assign each $P[2i]$ to either P_0 or P_1 uniformly at random, and assign $P[2i-1]$ to the other subsequence. By Proposition 2.4, we have

$$\mathrm{LIS}(P) \geq \mathbb{E}[\mathrm{LIS}(P_0) + \mathrm{LIS}(P_1)] - 2\,\mathrm{b}(P) \ .$$

Now apply a similar partitioning to each subsequence using independent coins, e. g., P_0 is split into P_{00} and P_{01}. Continue recursively in a similar fashion until we get a singleton subsequence P_σ for every $\sigma \in \{0,1\}^{\log n}$. For convenience, let ε denote the empty string, and define $P_\varepsilon = P$ and $\{0,1\}^0 = \{\varepsilon\}$.

Applying Proposition 2.4 recursively, we have

$$\mathrm{LIS}(P) \geq \sum_{\sigma \in \{0,1\}^{\log n}} \mathbb{E}[\mathrm{LIS}(P_\sigma)] - 2\sum_{k=1}^{\log n} \sum_{\sigma \in \{0,1\}^{k-1}} \mathbb{E}[\mathrm{b}(P_\sigma)] \ .$$

We rearrange this equation using the fact that if P_σ is a singleton sequence then $\mathrm{LIS}(P_\sigma) = 1$, and obtain

$$n - \mathrm{LIS}(P) \leq 2\,\mathbb{E}\left[\sum_{k=1}^{\log n} \sum_{\sigma \in \{0,1\}^{k-1}} \mathrm{b}(P_\sigma)\right] \ .$$

Notice that the sum $\sum_k \sum_\sigma \mathrm{b}(P_\sigma)$ in the right-hand side gets a contribution of 1 every time two characters $P[i], P[j]$ for which $i < j$ and $P[i] > P[j]$ become consecutive characters in a subsequence P_σ. Formally, for positions i and j such that $i < j$ and $P[i] > P[j]$, let the random variable Z_{ij} be the number of subsequences P_σ which contain $P[i], P[j]$ as consecutive characters. By linearity of expectation we get that

$$\frac{1}{2}\mathrm{ED}(P,Q) \leq n - \mathrm{LIS}(P) \leq 2 \sum_{i<j;P[i]>P[j]} \mathbb{E}[Z_{ij}] \ . \tag{2.3}$$

We claim that $\mathbb{E}[Z_{ij}] \leq 4/(j-i)$. To prove the claim, notice that $\mathbb{E}[Z_{ij}] = \sum_{z=1}^{\log n} \Pr[Z_{ij} \geq z]$ because Z_{ij} takes only integral values between 0 and $\log n$ (as $P[i]$ appears in $\log n$ subsequences). Therefore, it suffices to show for every integer $z \geq 1$ the upper bound

$$\Pr[Z_{ij} \geq z] \leq \frac{2^{2-z}}{j-i} \ .$$

Let us first show that $P[i], P[j]$ can become consecutive characters only after $\lfloor \log(j-i) \rfloor$ iterations (partitions of P). Indeed, if $j - i \geq 2$, then at the first iteration these two characters must belong to different blocks; thus, with probability $1/2$ the random partition of P sends them to different subsequences, in which case they will never form a consecutive pair of no P_σ, and with probability $1/2$ they are sent to the same subsequence, in which case the difference between their positions in that common subsequence is at least $\lfloor (j-i)/2 \rfloor$. Continuing similarly we see that if $2^l \leq j - i < 2^{l+1}$ then even after $l-1$ partitions, the difference between the positions of the two characters is at least 2, i. e., they can become consecutive only after l iterations. Since the different sequence partitions are independent, we get that

$$\Pr[Z_{ij} \geq 1] \leq \left(\frac{1}{2}\right)^{\lfloor \log(j-i) \rfloor} \leq \frac{2}{j-i} \ .$$

Similarly, if $P[i], P[j]$ are consecutive in a subsequence P_σ, they can be consecutive in a later iteration only if they are sent to the same subsequence of P_σ, which happens with proability $1/2$ or less. Fixing an integer $z \geq 1$ we can apply this argument $z-1$ times to get that $\Pr[Z_{ij} \geq z \mid Z_{ij} \geq 1] \leq (1/2)^{z-1}$. We thus conclude that $\Pr[Z_{ij} \geq z] \leq 2^{2-z}/(j-i)$, which completes the proof of the claim.

Using (2.3) together with the above claim regarding $\mathbb{E}[Z_{ij}]$, we get that

$$\frac{1}{2}\mathrm{ED}(P,Q) \leq 4 \sum_{i<j;P[i]>P[j]} \frac{1}{j-i} \ .$$

Notice that for every $i < j$ with $P[i] > P[j]$, the respective coordinates of $f(P)$ and $f(Q)$ are

$$f(P)_{\{P[i],P[j]\}} = \frac{1}{i-j} < 0 \qquad \text{and} \qquad f(Q)_{\{P[i],P[j]\}} = \frac{1}{P[i]-P[j]} > 0 \ ,$$

and hence $\left|f(P)_{\{P[i],P[j]\}} - f(Q)_{\{P[i],P[j]\}}\right| > 1/(j-i)$. We conclude that

$$\frac{1}{2}\,\mathrm{ED}(P,Q) \leq 4\|f(P) - f(Q)\|_1 \ ,$$

which completes the proof of Lemma 2.3. □

This completes the proof of Theorem 2.1. We note that the bounds in Lemma 2.2 and Lemma 2.3 are existentially tight, up to constant factors (for our embedding f).

3 Applications

We present several applications of our ℓ_1-embedding of the Ulam metric from Section 2. Following [2], we say that a string is *t-non-repetitive*, if all its t-substrings are distinct. We first extend the embedding to strings that are t-non-repetitive (Section 3.1). We also extend the above embedding to strings in which every character appears a bounded number of times (Section 3.2). Both extensions follow by showing that the corresponding strings can be embedded in the Ulam metric with low distortion. We then discuss the immediate applications of these embeddings to sketching algorithms (Section 3.3) and to Nearest Neighbor Search (Section 3.4).

A technically more involved application of the above embedding is an improvement to a result of Bar-Yossef et al. [2]. Call a string *(t,r)-non-repetitive* if every r successive t-substrings of it are distinct. We improve over the sketching algorithm of [2] for locally non-repetitive strings in two aspects: (i) we achieve an embedding result, which is stronger than a sketching algorithm (namely, a sketching algorithm follows quite easily); and (ii) we improve the approximation factor. See Section 3.5 for more details.

3.1 Embedding non-repetitive strings

Recall that a string is *t-non-repetitive*, if all its t-substrings are distinct. Let $X_{n,t}$ contain all the t-non-repetitive strings of length n over Σ.

Theorem 3.1. *The metric space $(X_{n,t}, \mathrm{ED})$ embeds with distortion $2t$ into the Ulam metric of dimension $n-t+1$ and alphabet size 2^t. Consequently, it embeds into ℓ_1 with distortion $O(t\log n)$.*

The proof of the first part of the theorem is based on a simple observation, and that of the second part is an immediate consequence of Theorem 2.1.

Proof. We define an embedding f of $(X_{n,t}, \mathrm{ED})$ into the aforementioned Ulam metric. First, we identify the Ulam metric alphabet $[2^t]$ with $\{0,1\}^t$ (using an arbitrary bijection). Now for a t-non-repetitive string $x \in \{0,1\}^n$, define $f(x)$ to be a length $n-t+1$ string (over $[2^t]$), whose jth coordinate is given by

$f(x)_j = b^{-1}(x[j]\dots x[j+t-1])$. To complete the proof, we will show that for every two strings $x,y \in \Sigma^n$,

$$\mathrm{ED}(x,y) \le \mathrm{ED}(f(x),f(y)) \le t\,\mathrm{ED}(x,y) \quad . \tag{3.1}$$

To prove the first inequality, consider a longest common subsequence between $f(x)$ and $f(y)$. Each character in it corresponds to a t-substring in x and in y. Taking the first character from each of these t-substrings, except for the last such t-substring, which is taken in its entirety, yields a subsequence that is common to x and y, and hence $\mathrm{LCS}(x,y) \ge \mathrm{LCS}(f(x),f(y))+t-1$. We obtain that

$$\frac{1}{2}\mathrm{ED}(x,y) \le n-\mathrm{LCS}(x,y) \le n-t+1-\mathrm{LCS}(f(x),f(y)) \le \mathrm{ED}(f(x),f(y)) \quad .$$

To prove the second inequality, fix a longest common subsequence of x and y, and with a slight abuse of notation let $\mathrm{LCS}(x,y)$ denote both this sequence and its length. Consider all the t-substrings of x and of y that are entirely contained in $\mathrm{LCS}(x,y)$. Observe that the number of such t-substrings is at least $n-t+1-t(n-\mathrm{LCS}(x,y))$, because each of the $n-\mathrm{LCS}(x,y)$ characters that do not belong to $\mathrm{LCS}(x,y)$ participates in t or fewer t-substrings. These t-substrings in x and in y give rise to a subsequence that is common to $f(x)$ and $f(y)$. Therefore, $\mathrm{LCS}(f(x),f(y)) \ge n-t+1-t(n-\mathrm{LCS}(x,y))$, implying that

$$\mathrm{ED}(f(x),f(y)) \le 2(n-t+1-\mathrm{LCS}(f(x),f(y))) \le t(n-\mathrm{LCS}(x,y)) \le t\,\mathrm{ED}(x,y) \quad .$$

□

3.2 Embedding bounded-occurrence strings

We say that a string P has *t-bounded-occurrence* if every character $a \in \Sigma$ appears at most t times in P. Let $B_{n,t} \subseteq \Sigma^n$ contain all the t-bounded-occurrence strings of length n over Σ.

Theorem 3.2. *The metric space $(B_{n,t},\mathrm{ED})$ embeds with distortion t into the Ulam metric of dimension n over an extended alphabet of size $t|\Sigma|$. Consequently, it embeds into ℓ_1 with distortion $O(t\log n)$.*

The proof the first part of the theorem is based on a simple observation, and that of the second part follows immediately from Theorem 2.1.

Proof. Let Σ' be an alphabet of size $t|\Sigma|$, and associate every character $a \in \Sigma$ with t distinct characters $a_1,\dots,a_t \in \Sigma'$. Given a string x, let $f(x)$ be the string obtained from x by replacing, for every $j \in [t]$ and every character $a \in \Sigma$, the jth occurrence of a in x with the character $a_j \in \Sigma'$. To complete the proof of the first part, it would suffice to prove that for every two strings $x,y \in \Sigma^n$,

$$\frac{1}{2}\mathrm{ED}(x,y) \le \mathrm{ED}(f(x),f(y)) \le t\,\mathrm{ED}(x,y) \quad , \tag{3.2}$$

and it is indeed straightforward to verify this inequality. □

3.3 Sketching algorithms

A sketching algorithm for edit distance consists of two procedures that work in concert as follows. The first procedures produce a fingerprint, called *sketch*, from each of the input strings, and the second procedure uses solely the sketches to approximate the edit distance between the two strings. In the k vs. m gap version of the problem, there is a promise that the edit distance between the two strings is either at most k or more than m, and we wish to decide which of the two cases holds. The key feature is that the sketch of each string is constructed without knowledge of the other string. The procedures are randomized and are allowed to share random coins, and the main measure of complexity is the size of the sketches produced. For actual applications it is also desirable that both procedures are efficient (say run in time that is polynomial in their input size).

In contrast to Hamming distance, whose sketching complexity is well-understood [15, 18, 10], relatively little is known about sketching of edit distance. The result of Ostrovsky and Rabani [20] gives a sketching algorithm that, for every $k = k(n)$, distinguishes between pairs of strings at edit distance at most k and at least $k \cdot 2^{O(\sqrt{\log n \log\log n})}$ using sketches of size $O(1)$.

For strings that are t-non-repetitive (including e. g. permutations), Bar-Yossef et al. [2] give a sketching algorithm that solves, for every $k = k(n)$, the k vs. $\Omega(tk^2)$ gap edit distance problem using sketches of size $O(1)$. We improve over their algorithm as follows.

Theorem 3.3 (Sketching t-non-repetitive strings). *For every $k = k(n)$ there exists a polynomial-time sketching algorithm that solves the k vs. $\Omega(kt \log n)$ gap edit distance problem on t-non-repetitive strings of length n using sketches of size $O(1)$.*

The proof of the theorem is a simple consequence of the ℓ_1-embedding from Theorem 3.1, First, convert the ℓ_1-metric can be into a scaled Hamming metric. Observe that a scaling factor of $O(n^2)$ suffices: rounding each coordinate in the ℓ_1-embedding to multiples of $1/Cn^2$, for a sufficiently large constant $C > 0$, increases the distortion by a factor 2, because $f(P)_{\{a,b\}} - f(Q)_{\{a,b\}}$ for f in (2.1) is either zero or has absolute value $\Omega(1/n^2)$. (A different argument is that $f(P)$ has at most n^2 nonzero coordinates, and thus the total error due to rounding is at most additive $1/C$.) Then, use the sketching algorithm stated below for Hamming spaces, which is implicit in [18], and can also be derived from the locality-sensitive hashing algorithms of [15, 13] (using the fact that for binary strings there is a direct correspondence between Hamming distance and ℓ_2 distance).

Theorem 3.4 (Sketching Hamming metric [18]). *For every $\varepsilon > 0$ and $k = k(n)$, there is a polynomial-time sketching algorithm that solves the k vs. $(1+\varepsilon)k$ gap Hamming distance problem in binary strings of length n, with a sketch of size $O(1/\varepsilon^2)$.*

We note that besides being a very basic computational primitive for massive data sets (see e. g. [5, Section 4.6]), sketching is also related to (i) Nearest Neighbor Search (see below), (ii) protocols that are secure (i. e., leak no information), cf. [10], and (iii) the simultaneous messages communication model with public coins [21].

3.4 Nearest Neighbor Search

One of the most extensively studied computational problems is *Nearest Neighbor Search* (NN): Given a set S of points in a metric space X, preprocess S so as to efficiently answer queries for finding the point in S that is closest to a query point $q \in X$. The last decade has seen the advent of data structures for approximate NN in (high) n-dimensional normed spaces. In particular, algorithms with preprocessing space (storage) polynomial in n and $|S|$ and query time that is polynomial in n and $\log|S|$, were designed in [15, 18] for ℓ_1^n and ℓ_2^n (achieving approximation factor $1+\varepsilon$ for every constant $\varepsilon > 0$) and in [12] for ℓ_∞^n (achieving approximation factor $O(\log\log n)$). Other algorithms for ℓ_1^n and ℓ_2^n, due to [15, 11], achieve $1+\varepsilon$ approximation require preprocessing space that is more moderate (subquadratic in $|S|$) and have query time that is sublinear in $|S|$ (roughly $O(|S|^{1/(1+\varepsilon)})$).

In contrast, the known algorithmic guarantees for approximate NN in edit distance metrics (X, ED) are much weaker. For the general case (Σ^n, ED), Indyk [14] designed, for every fixed $\alpha > 0$, a constant factor approximation (exponential in $1/\alpha$) with space requirement that is exponential in n^α. The recent embedding result of Ostrovsky and Rabani [20] leads to a nearest neighbor data structure with approximation factor $2^{O(\sqrt{\log n \log\log n})}$ and polynomial space requirement.

Combining our ℓ_1-embedding from Theorem 3.2 with these NN algorithms for ℓ_1 (or Hamming space or ℓ_2, as discussed in Section 3.3) immediately yields a nearest neighbor algorithm for t-non-repetitive strings (including e. g. permutations) achieving approximation factor $O(t \log n)$, which improves over the bound obtain in [2] for this case. In particular, using the algorithms of [15, 18] results in query time $(n + \log|S|)^{O(1)}$, and space requirement that $(n+|S|)^{O(1)}$. Similarly, using the algorithms of [15, 11] results in query time that is sublinear in $|S|$ and space requirement that is subquadratic in $|S|$.

The sketching algorithm can alternatively be used to speed up the naive algorithm that computes the edit distance between the query q and every data point $x \in S$. Simply replace each edit distance computation with an estimate derived from a sketch of x (computed at the preprocessing stage) and a sketch of q. The number of iterations would still be $O(|S|)$, but each iteration will be much faster–about $O(\log|S| + \log\log n)$ time.

3.5 Embedding locally non-repetitive strings

We can further generalize Theorem 3.1 and obtain an embedding of strings that are locally non-repetitive (see the definition below). The guarantee of this embedding is slightly weaker than a low distortion, since it approximates well only distances that are sufficiently small. An interesting aspect of our embedding is that it uses the sketching algorithm of [18] to obtain an embedding; this is opposite to the usual direction, where an embedding is used to obtain a sketching algorithm.

Our results improve over [2] in several respects: (i) We give an embedding, which consequently leads to a sketching algorithm, while [2] only give a sketching algorithm. An embedding result is stronger (unless it is computationally inefficient) since it has to simultaneously handle exponentially many strings, including pairs x, y with rather different distance $\mathrm{ED}(x,y)$, while a sketching algorithm is only required to have a high success probability for every pair. (ii) Our approximation factor is smaller since we rely on the Ulam metric embedding.

Definition 3.5 (Locally non-repetitive string). A string is called (t,r)*-non-repetitive*, or in short *locally non-repetitive*, if every r successive t-substrings are distinct, i. e., for each interval $\{i, \dots, i+r-1\}$, the r substrings of length t each that start in this interval are distinct.

Let $X_{n,t,r}$ contain all the (t,r)-non-repetitive strings of length n over Σ. Notice that this family contains $X_{n,t}$ since every t-non-repetitive string is also (t,r)-non-repetitive.

Theorem 3.6 (Embedding). *For every $t = t(n)$ and every $k = k(n)$, there exists an embedding f of the $(t, 180tk)$-non-repetitive strings into ℓ_1, such that for every two such strings x, y,*

$$\Omega(\min\{k, \mathrm{ED}(x,y)/t\log(tk)\}) \le \|f(x) - f(y)\| \le \mathrm{ED}(x,y) \ .$$

This embedding builds on the iterative anchors technique used in the sketching algorithm of Bar-Yossef et al. [2, Section 2]. The basic idea is to pick a set of non-overlapping substrings of length t in a coordinated fashion. These substrings are referred to as anchors and partition the string into disjoint substrings. The substrings between anchors are embedded into ℓ_1 and the embedding for the original string is obtained by combining these embeddings in a suitable way. The key idea from [2] is a method to pick these anchors in such a way that if x and y are two strings with small edit distance, then the anchor selection process picks the same anchors for both x and y. One technical difference is that, between successive anchors, we employ the ℓ_1-embedding from Section 2. Another difference is that as a final step we apply the ℓ_1 sketching algorithm of [18], which effectively "thresholds" the ℓ_1 distance between images. For clarity, we make no attempt to optimize constants.

Proof. We describe the embedding of a $(t, 180tk)$-non-repetitive strings x using a randomized procedure that generates a bit $f'(x)$. The embedding f into ℓ_1 will then be the concatenation of $f'(x)$, ranging over all possible outcomes for the coin tosses, with suitable scaling.

Fix $W := 56tk$. Augment the alphabet Σ with $2W + t$ new characters $a_1, \ldots, a_{2W+t}$ and append to x the fixed string $a_1 \ldots a_{2W+t}$. Select a sequence of disjoint substrings $\alpha_1, \ldots, \alpha_{r_x}$ of x, called "anchors," iteratively as follows. Maintain a sliding window of length $2W + t$ over the string x. The left endpoint of the sliding window is denoted by c; initially, c is set to 1. At each step, say step i, consider the W substrings of length t whose starting position lies in the interval $[c + W \ldots c + 2W - 1]$, and denote the jth such substring, for $j \in [W]$, by

$$s_{i,j} = x[c + W + j - 1 \ldots c + W + j + t - 2] \ .$$

Select at random a permutation Π_i of Σ^t, and set the anchor α_i to be a substring $s_{i,l}$ that is minimal according to Π_i (breaking ties arbitrarily), i. e.,

$$\Pi_i(s_{i,l}) = \min\{\Pi_i(s_{i,1})), \ldots, \Pi_i(s_{i,W})\} \ .$$

Then slide the window by setting c to the position immediately following the anchor, i. e.,

$$c \leftarrow c + W + l + t - 1 \ .$$

If this new value of c is at most n start a new iteration. Otherwise, stop, letting $r_x \le O(n/tk)$ be the number of anchors collected.

For $i \in [r_x]$, let $\phi^i = \phi^i(x)$ be the substring of x starting at the position immediately after the last character of anchor α_{i-1} and ending at the last character of α_i. By convention, ϕ^1 starts at position 1.

Now embed each ϕ^i into $\ell_1^{O(tk)}$ using Theorem 3.1; notice that ϕ^i is a substring of x of length at most $2W + t \le 180tk$, and is thus t-non-repetitive. Next concatenate the resulting $r_x = O(n/tk)$ images into a vector $\varphi(x) \in \ell_1^{O(n)}$ (appending 0's as necessary). Finally, choose a random bit string $r \in \{0,1\}^{O(n)}$ of the same length, such that for all i, $r_i = 1$ independently with probability $1/kt\log(kt)$, and let

$$f'(x) = \sum_i r_i \cdot \varphi_i(x) \mod 2 \ .$$

(This step is similar to [18], though the purpose of using it here is very different.)

The embedding's correctness follows immediately from the next two lemmas, which we shall prove shortly.

Lemma 3.7. *If x and y are $(t, 180tk)$-non-repetitive strings then*

$$\Pr[f'(x) \neq f'(y)] \leq O(\mathrm{ED}(x,y)/k) \ .$$

Lemma 3.8. *If x and y are $(t, 180tk)$-non-repetitive strings then*

$$\Pr[f'(x) \neq f'(y)] \geq \Omega(\min\{\mathrm{ED}(x,y)/kt\log(kt), 1\}) \ .$$

The proof of Theorem 3.6 is completed by observing that the concatenation of bits f' with suitable scaling yields an embedding f which satisfies

$$\|f(x) - f(y)\|_1 = k\,\mathbb{E}\,|f'(x) - f'(y)| = k\Pr[f'(x) \neq f'(y)] \ .$$

$\square$

Proof of Lemma 3.7. The preliminary step of appending x and y with the same string of length $2W + t$ clearly does not change $\mathrm{ED}(x,y)$. Now fix a sequence of edit operations τ that transforms this new x into the new y and uses $\mathrm{ED}(x,y)$ edit operations. Let α_i be the ith anchor chosen for x and let β_i be the ith anchor chosen for y. Let $r = \min\{r_x, r_y\}$. As in the proof of Lemma 2.6 in [2], with probability at least $1 - \mathrm{ED}(x,y)/7k$, the following event happens: for all $i \in [r]$, the tranformation τ (sequence of edit operations) maps α_i to β_i with no edit operations inside α_i or β_i.[4] If this happens, we say that the *anchors match*.

Assume for the moment that the anchors match. Using the fact that $\{\phi^i(x)\}_{i\in[r]}$ are disjoint substrings of x and similarly $\{\phi^i(y)\}_{i\in[r]}$ for y, we get that

$$\mathrm{ED}(x,y) = \sum_{i\in[r]} \mathrm{ED}(\phi^i(x), \phi^i(y)) \ .$$

Furthermore, at least one of the anchors α_r and β_r is the last anchor in its string and thus contains one of the unique $2W + t$ characters that were appended to x and y. But since the anchors $\alpha_r = \beta_r$, both must be the last anchor in their string, and thus $r_x = r_y$. Using the guarantees of Theorem 3.1, we get that

$$\|\varphi(x) - \varphi(y)\|_1 \leq \sum_{i\in[r]} O(t\log(2W + tk))\,\mathrm{ED}(\phi^i(x), \phi^i(y)) \leq O(t\log(tk))\,\mathrm{ED}(x,y) \ .$$

[4] The argument in [2] goes roughly as follows: at a single iteration, the probability that the anchor selection goes wrong is at most t times the number of edit operations inside the current window divided by W (since there are W choices for the anchor). It can be verified that an edit operation can only affect one iteration, and a union bound over all iterations gives an upper bound of $\mathrm{ED}(x,y)/7k$.

Finally, the analysis in [18] of a random inner product modulo 2 shows that

$$\Pr\left[f'(x) \neq f'(y) \mid \varphi(x), \varphi(y)\right] = \Theta(\min\{\|\varphi(x) - \varphi(y)\|/kt\log(kt), 1\}) \ . \tag{3.3}$$

We can then bound $\Pr[f'(x) \neq f'(y)]$ from above by conditioning on whether the anchors match. If they do match, $f'(x) \neq f'(y)$ with probability $O(\mathrm{ED}(x,y)/k)$. Otherwise (which happens with probability at most $\mathrm{ED}(x,y)/7k$), $f'(x) \neq f'(y)$ with probability at most 1. We thus conclude that

$$\Pr[f'(x) \neq f'(y)] \leq O(\mathrm{ED}(x,y)/k) \ .$$

□

Proof of Lemma 3.8. The preliminary step of appending x and y with the same string of length $2W+t$ clearly does not change $\mathrm{ED}(x,y)$. Now similar to the proof of Lemma 2.8 in [2], let $r = \max\{r_x, r_y\}$ and for $i = r_x+1, \ldots, r$ let $\phi^i(x) = \varepsilon$ be the empty string and similarly for $i = r_y+1, \ldots, r$ let $\phi^i(y) = \varepsilon$. Let g be the ℓ_1-embedding from Theorem 3.1. Then for all $i \in [r]$ we have

$$\|g(\phi^i(x)) - g(\phi^i(y))\| \geq \Omega(\mathrm{ED}(\phi^i(x), \phi^i(y))) \ .$$

Since the substrings $\{\phi^i(x)\}_{i\in[r]}$ induce a partition of the string x and similarly $\{\phi^i(y)\}_{i\in[r]}$ for y, we get that

$$\mathrm{ED}(x,y) \leq \sum_{i\in[r]} \mathrm{ED}(\phi^i(x), \phi^i(y)) \leq O(\|\varphi(x) - \varphi(y)\|) \ .$$

The lemma follows by applying the analysis of a random inner product modulo 2, as stated in (3.3). □

Improved sketching algorithm The embedding of Theorem 3.6 leads to the following sketching result.

Theorem 3.9 (Sketching). *For every $t = t(n)$ and every $k = k(n)$, there exists a polynomial-time efficient sketching algorithm that solves the k vs. $\Omega(tk\log k)$ gap edit distance problem for $(t, 180tk)$-non-repetitive strings using sketches of size $O(1)$.*

The proof follows in a straightforward way by "concatenating" the embedding of Theorem 3.6 with a sketching algorithm for the k' vs. $k'(1+\varepsilon)$ gap Hamming distance (or ℓ_1) problem that is implicit in [15, 18]. We note that the embedding uses many dimensions (coordinates), but for the purpose of sketching it suffices to generate only $O(kt\log(kt))$ coordinates f' at random, which can be done efficiently using the shared random coins. It is also easy to verify that the random permutation Π can be replaced by an almost min-wise hash family that is efficiently computable using shared randomness, similar to [2].

Note that a permutation is $(1,r)$-non-repetitive for every $r \geq 1$, and so this theorem offers a somewhat unexpected small improvement for the Ulam metric (over Theorem 3.3), reducing the gap from $O(\log n)$ to $O(\log k)$.

Acknowledgements We thank the anonymous reviewers for helpful comments.

References

[1] * D. ALDOUS AND P. DIACONIS: Longest increasing subsequences: from patience sorting to the Baik-Deift-Johansson theorem. *Bull. Amer. Math. Soc. (N.S.)*, 36(4):413–432, 1999. [BullAMS:1999-36-04/S0273-0979-99-00796-X]. 1

[2] * Z. BAR-YOSSEF, T. S. JAYRAM, R. KRAUTHGAMER, AND R. KUMAR: Approximating edit distance efficiently. In *Proc. 45th FOCS*, pp. 550–559. IEEE Computer Society Press, 2004. [FOCS:10.1109/FOCS.2004.14]. 1.1, 3, 3.3, 3.4, 3.5, 3.5, 3.5, 4, 3.5, 3.5

[3] * T. BATU, F. ERGÜN, J. KILIAN, A. MAGEN, S. RASKHODNIKOVA, R. RUBINFELD, AND R. SAMI: A sublinear algorithm for weakly approximating edit distance. In *Proc. 35th STOC*, pp. 316–324. ACM Press, 2003. [STOC:10.1145/780542.780590]. 1.2

[4] * J. BOURGAIN: On Lipschitz embedding of finite metric spaces in Hilbert space. *Israel J. Math.*, 52(1-2):46–52, 1985. 2

[5] * G. CORMODE: *Sequence Distance Embeddings*. PhD thesis, University of Warwick, 2003. 1, 1, 3.3

[6] * G. CORMODE AND S. MUTHUKRISHNAN: The string edit distance matching problem with moves. In *Proc. 13th Ann. ACM–SIAM Symp. on Discrete Algorithms (SODA'02)*, pp. 667–676, 2002. [SODA:545381.545470]. 1.2

[7] * G. CORMODE, S. MUTHUKRISHNAN, AND S. C. SAHINALP: Permutation editing and matching via embeddings. In *Proc. 28th Internat. Coll. on Automata, Languages, and Programming (ICALP'01)*, volume 2076 of *Lecture Notes in Computer Science*, pp. 481–492. Springer, 2001. [ICALP:hf0vwuh0rcyujug1]. 1, 1.1, 1.2, 2

[8] * G. CORMODE, M. PATERSON, S. C. SAHINALP, AND U. VISHKIN: Communication complexity of document exchange. In *Proc. 11th Annual ACM–SIAM Symp. on Discrete Algorithms (SODA'00)*, pp. 197–206, 2000. [SODA:338219.338252]. 1.2

[9] * P. ENFLO: On the nonexistence of uniform homeomorphisms between L_p-spaces. *Ark. Mat.*, 8:103–105, 1969. 1

[10] * J. FEIGENBAUM, Y. ISHAI, T. MALKIN, K. NISSIM, M. J. STRAUSS, AND R. N. WRIGHT: Secure multiparty computation of approximations. In *Proceedings of 28th International Colloquium on Automata, Languages, and Programming*, volume 2076 of *Lecture Notes in Computer Science*, pp. 927–938. Springer, 2001. [ICALP:cpq5t97vrymq7q3n]. 3.3, 3.3

[11] * A. GIONIS, P. INDYK, AND R. MOTWANI: Similarity search in high dimensions via hashing. In *Proc. 25th Internat. Conf. on Very Large Data Bases*, pp. 518–529. Morgan Kaufmann Publishers Inc., 1999. [VLDB:645925.671516]. 3.4

[12] * P. INDYK: On approximate nearest neighbors in non-euclidean spaces. In *Proc. 39th FOCS*, pp. 148–155. IEEE Computer Society Press, 1998. [FOCS:10.1109/SFCS.1998.743438]. 3.4

[13] * P. INDYK: Dimensionality reduction techniques for proximity problems. In *Proc. 11th Ann. ACM-SIAM Symp. on Discrete Algorithms (SODA'00)*, pp. 371–378. SIAM, 2000. [SODA:338219.338582]. 3.3

[14] * P. INDYK: Approximate nearest neighbor under edit distance via product metrics. In *Proc. 15th Ann. ACM–SIAM Symp. on Discrete Algorithms*, pp. 646–650. SIAM, 2004. [SODA:982792.982889]. 1.1, 3.4

[15] * P. INDYK AND R. MOTWANI: Approximate nearest neighbors: towards removing the curse of dimensionality. In *30th STOC*, pp. 604–613. ACM Press, 1998. [STOC:10.1145/276698.276876]. 3.3, 3.3, 3.4, 3.5

[16] * S. KHOT AND A. NAOR: Nonembeddability theorems via Fourier analysis. *Mathematische Annalen*, 334(4):821–852, 2006. [Springer:n4671147n1684344]. 1

[17] * R. KRAUTHGAMER AND Y. RABANI: Improved lower bounds for embeddings into L_1. In *Proc. 16th Ann. ACM–SIAM Symp. on Discrete Algorithms*, pp. 1010–1017. SIAM, 2006. [SODA:1109557.1109669]. 1

[18] * E. KUSHILEVITZ, R. OSTROVSKY, AND Y. RABANI: Efficient search for approximate nearest neighbor in high dimensional spaces. *SIAM Journal on Computing*, 30(2):457–474, 2000. [SICOMP:30/34717]. 3.3, 3.3, 3.4, 3.4, 3.5, 3.5, 3.5, 3.5

[19] * S. MUTHUKRISHNAN AND S. C. SAHINALP: Approximate nearest neighbors and sequence comparisons with block operations. In *Proc. 32nd STOC*, pp. 416–424. ACM Press, 2000. [STOC:10.1145/335305.335353]. 1.2

[20] * R. OSTROVSKY AND R. RABANI: Low distortion embeddings for edit distance. In *Proc. 37th STOC*, pp. 218–224. ACM Press, 2005. [STOC:1060590.1060623]. 1, 1.1, 1.1, 3.3, 3.4

[21] * A. C-C. YAO: Some complexity questions related to distributive computing. In *Proc. 11th STOC*, pp. 209–213. ACM Press, 1979. [STOC:10.1145/800135.804414]. 3.3

AUTHORS

Moses Charikar
Dept. of Computer Science
Princeton University
35 Olden Street
Princeton, NJ 08540, USA
moses@cs.princeton.edu
http://www.cs.princeton.edu/~moses/

Robert Krauthgamer
IBM Almaden Research Center
Department K53/B2
650 Harry Road
San Jose, CA 95120, USA
robi@almaden.ibm.com
http://www.almaden.ibm.com/cs/people/robi/

ABOUT THE AUTHORS

MOSES CHARIKAR is an Assistant Professor in the Computer Science department at Princeton University. He received his Ph. D. in 2000 from Stanford University under the supervision of Rajeev Motwani. Before that, he obtained his undergraduate degree from the Indian Institute of Technology, Bombay. His research interests are in approximation algorithms, metric embeddings, and algorithmic techniques for large data sets. His work on dimension reduction in ℓ_1 won the Best Paper award at FOCS 2003. A one year stint in the research group at Google gave him an opportunity to apply his theoretical ideas in the real world. He still reaps the benefits of that experience – he has successfully managed to retain the top spot for a Google search on his last name, but has wisely given up trying to compete with his well-known namesake for searches on his first name.

ROBERT KRAUTHGAMER is a Research Staff Member in the theory group at the IBM Almaden Research Center in San Jose, CA. He received his Ph. D. in 2001 from the Weizmann Institute of Science in Israel. A paper, coauthored (as part of his thesis) with his advisor, Uri Feige, was awarded the 2005 SIAM Outstanding Paper Prize. Subsequently he spent two years as a postdoc in the theory group at Berkeley. His research interests include combinatorial algorithms, finite metric spaces, high-dimensional geometry, data analysis, and related areas. His favorite sport since youth has been swimming; once he swam across the Sea of Galilee in a 10km competitive race, and was the last one to arrive at the finish line.