Overcoming the ℓ_1 Non-Embeddability Barrier: Algorithms for Product Metrics

Alexandr Andoni
MIT
andoni@mit.edu

Piotr Indyk
MIT
indyk@mit.edu

Robert Krauthgamer*
Weizmann Institute of Science
robert.krauthgamer@weizmann.ac.il

Abstract

A common approach for solving computational problems over a difficult metric space is to embed the "hard" metric into L_1, which admits efficient algorithms and is thus considered an "easy" metric. This approach has proved successful or partially successful for important spaces such as the edit distance, but it also has inherent limitations: it is provably impossible to go below certain approximation for some metrics.

We propose a new approach, of embedding the difficult space into richer host spaces, namely iterated products of standard spaces like ℓ_1 and ℓ_∞. We show that this class is *rich* since it contains useful metric spaces with only a constant distortion, and, at the same time, it is *tractable* and admits efficient algorithms. Using this approach, we obtain for example the first nearest neighbor data structure with $O(\log \log d)$ approximation for edit distance in non-repetitive strings (the Ulam metric). This approximation is exponentially better than the *lower bound* for embedding into L_1. Furthermore, we give constant factor approximation for two other computational problems. Along the way, we answer positively a question posed in [Ajtai, Jayram, Kumar, and Sivakumar, STOC 2002]. One of our algorithms has already found applications for smoothed edit distance over 0-1 strings [Andoni and Krauthgamer, ICALP 2008].

1 Introduction

An *embedding* is a mapping from one metric space (the "guest" space) to another (the "host" space), which preserves the distances between every pair of points, up to a multiplicative factor called *the distortion*. Embeddings provide a general method for solving problems over "hard" metric spaces, by embedding them into "easy" ones. Since the mid 1990's, an extensive body of work has been devoted to this method, resulting in many efficient approximation algorithms for a wide variety of problems [26, 39, 42, 30].

One of the most convenient host spaces discovered so far is the ℓ_1 space, i.e., the standard real vector space under the ℓ_1-norm. This is because (a) it is rich – many interesting and useful spaces can be embedded into it with low distortion, and (b) it is tractable – several computational problems in it admit efficient algorithms. Notable successes of this approach of obtaining algorithms via embedding into ℓ_1 include, among others, approximation algorithm for the sparsest-cut in a graph [40, 12, 11, 10], approximate nearest neighbor search and sketching under edit distance [44, 16] and under Earth-Mover Distance [15, 32, 43, 7].

However, it was recently discovered that this approach has inherent limitations. In particular, for the aforementioned problems, embedding into ℓ_1 cannot result in algorithms with constant approximation factors [37, 36, 38, 20, 43, 8, 7]. Thus, to make progress on those problems via the embedding approach, we must identify richer, yet tractable, classes of host spaces.

We undertake precisely this task. We focus mostly on a specific variant of a metric that has found many applications, namely edit distance. Our results include greatly improved algorithms for three different problems. However, our contribution should be seen from a bigger perspective: we propose a generic approach that may be useful in many contexts, and provide an implementation for one concrete metric as a proof-of-concept. Indeed, another metric important for applications is the Earth-Mover Distance, and our partial results for it indicate that this new approach may be applicable here too. In both cases, embedding into ℓ_1, or even into related spaces such as ℓ_2-squared, provably requires high distortion.

Our approach is to consider a class of alternative host spaces, namely the *iterated products* of standard spaces like ℓ_1 and low-dimensional ℓ_∞. These spaces exhibit a better balance between *richness* and *tractability*. Indeed, we identify a sweet spot: the spaces are rich enough to accommodate the intended guest spaces with only a constant distortion, while admitting quite efficient algorithms. As a result, we obtain several improved algorithms that achieve approximation factors that are at least exponentially better than what is possible via the ℓ_1 approach. Along the way we answer positively several questions left open in [2, 28].

An example of our host spaces is the space

$$\bigoplus_{(\ell_2)^2}^{k} \bigoplus_{\ell_\infty}^{l} \ell_1^m$$

*Part of this work was done while at IBM Almaden. This research was supported in part by The Israel Science Foundation (grant #452/08).

which reads as follows. For a metric space $\mathcal{M}$, an integer k, and $p \geq 1$, the ℓ_p-*product* of k copies of $\mathcal{M}$, denoted $\bigoplus_{\ell_p}^k \mathcal{M}$, is the space $\mathcal{M}^k$ endowed with the distance function $d_{p,\mathcal{M}}(x,y) = \big(\sum_{i=1}^k d_{\mathcal{M}}(x_i,y_i)^p\big)^{1/p}$. The definition naturally extends to ℓ_∞ products and squared-ℓ_2 products. Then, the double iterated product space $\bigoplus_{(\ell_2)^2}^k \bigoplus_{\ell_\infty}^l \ell_1^m$ contains points $x \in \mathbb{R}^{k \cdot l \cdot m}$ with the distance function

$$d_{\text{NEG},\infty,1}(x,y) = \sum_{a=1}^{k} \Big(\max_{b=1,2,\ldots,l} \Big\{ \sum_{c=1}^{m} |x_{a,b,c} - y_{a,b,c}| \Big\} \Big)^2,$$

where $x_{a,b,c}$ stands for coordinate $(a-1)lm + (b-1)m + c$ of x.

1.1 Two Hard Metrics

Ulam metric. The edit distance (aka *Levenshtein distance*) between two strings x and y, denoted $\mathrm{ed}(x,y)$, is the minimum number of character insertions, deletions, and substitutions needed to transform one string into the other. This distance is of key importance in several fields such as computational biology and text processing, and consequently computational problems involving the edit distance were studied quite extensively. Throughout, we consider strings of length d over an alphabet Σ.

The Ulam metric is a specialization of edit distance to non-repetitive strings, where a string is *non-repetitive* if every symbol appears at most once in it. There are several motivations for studying this variant. From a practical perspective, strings with limited or no repetitions appear in several important contexts, e.g. ranking of objects such as webpages (see e.g. [2, Sections 1.2 and 6] and [41]). In fact, our motivation is similar to [2], but we study a different distance function on non-repetitive strings (which they explicitly mention as open).

From a theoretical point of view, Ulam metric appears to retain one of the core difficulties of the edit distance on general strings, namely the existence of "misalignments" between the two strings. In fact, there is no known lower bound that would strictly separate general edit distance from Ulam: all known lower bounds are nearly the same (quantitatively) for both metrics. In particular, embedding the Ulam metric and the edit distance on general strings into ℓ_1 and similar spaces such as ℓ_2-squared requires distortion $\Omega(\frac{\log d}{\log\log d})$ [8]. Thus, the Ulam metric is a concrete roadblock that we must overcome if we ever hope to obtain efficient algorithms for the general edit distance with strongly sub-logarithmic approximation.

Earth-Mover Distance (EMD). Every point in this metric is a distribution π supported on the two-dimensional grid $[d]^2$, and the distance between π_A, π_B is given by the transportation cost between these two distributions with respect to ℓ_1 distance in the plane. For example, if the two distributions are uniform over equal-size multisets $A, B \subset [d]^2$, respectively, then $\mathrm{EMD}(\pi_A, \pi_B)$ is the minimum cost bipartite matching between the elements in A and in B. This metric has several applications, most notably in computer vision [45]. Embedding EMD into ℓ_1 requires distortion $\Omega(\sqrt{\log d})$ and even into ℓ_2-squared it is $\omega(1)$ [43].

1.2 Problem Definitions and Our Results

We show how our approach of using product metrics leads to a suite of new algorithms for the Ulam metric. We also give a new NNS algorithm for EMD. In all cases, the approximation ratios we achieve are exponentially better than previously known bounds, albeit sometimes at the expense of some loss in other parameters. Moreover, our bounds overcome (or bypass) the ℓ_1 non-embeddability barrier. We proceed to describe our algorithmic results, which are also summarized in Table 1.

Nearest Neighbor Search (NNS). A major algorithmic challenge is the problem of Nearest Neighbor Search (NNS) under various metrics. In this problem, we wish to design a data structure that preprocesses a dataset of n points, so that when a query point is given, the database reports query's nearest neighbor (i.e., a point in the dataset with the smallest distance to the query point). A ρ-approximate NNS algorithm reports a point whose distance from the query is at most ρ times that of the nearest neighbor. In many cases, the challenge is to use space (storage) that is *polynomial* in n and in the point length d, and query time that is *strongly sublinear* in n, namely $n^\varepsilon d^{O(1)}$ for an arbitrarily small constant $\varepsilon > 0$. We shall call such a data structure *efficient*.

We devise for the Ulam metric an NNS scheme that achieves $O(\varepsilon^{-3}\log\log d)$-approximation using $n^\varepsilon d^{O(1)}$ query time and $(dn)^{O(1)}$ preprocessing, for any desired constant $\varepsilon > 0$. Our approximation factor is exponentially better than the $O(\log d)$ previously known via ℓ_1-embedding, due to [16], although their query time is better (logarithmic in n). In fact, our result is the first algorithm that beats the ℓ_1 distortion lower bound of $\Omega(\log d / \log\log d)$ [8].

Sketching Algorithms. Another important algorithmic primitive is the communication complexity of distance estimation, and more specifically in the sketching model. The sketch of a point x is a (randomized) mapping of x into a short "fingerprint" $\mathbf{sk}(x)$, such that sketches of two points, $\mathbf{sk}(x)$ and $\mathbf{sk}(y)$, are sufficient to distinguish (with high probability) between the case where x, y are at distance $d(x,y) \leq R$, and the case where $d(x,y) > \rho R$, for an approximation factor $\rho > 1$ and a parameter $R \in \mathbb{R}^+$. The main parameter of a sketching algorithm is its *sketch size*, the bit length of $\mathbf{sk}(x)$.

The sketching model is viewed as a basic computational primitive in massive data sets [3, 13, 22]. For example, constant-size sketches for an approximation ρ imply efficient NNS with approximation $(1+\varepsilon)\rho$ for every fixed $\varepsilon > 0$.

We design a sketching algorithm for Ulam metric that achieves $O(1)$-approximation using sketch size $\log^{O(1)} d$. Again, this approximation is a significant improvement

	Problem	Reference	Approximation	Comments
Ulam	**NNS**[a]	[16]	$O(\log d)$	
		[28]	$3^{\alpha-1}$	space $n^{O(d^{1/\alpha})}$
	(lower bound)	[8]	$\Omega(\log d/\log\log d)$	for embedding into ℓ_1, $(\ell_2)^2$
		This paper	$O(\log\log d)$	query time $d^{O(1)}n^{\varepsilon}$
	Sketching	[16]	$O(\log d)$	sketch size $O(1)$
	(lower bound)	[8]	$O(1)$	sketch size $\Omega(\log\log d)$
		This paper	$O(1)$	sketch size $(\log d)^{O(1)}$
	Distance Estimation	[14]	d^{ε} (w/restrictions)	$\tilde{O}(d^{1-2\varepsilon})$ running time
	(lower bound)	[14]	$O(1)$	$\Omega(\sqrt{\mathrm{ed}(P,Q)} + d/\mathrm{ed}(P,Q))$
		This paper	$O(1)$	$\tilde{O}(d/\sqrt{\mathrm{ed}(P,Q)})$ running time
EMD	**NNS**	[15, 32]	$O(\log d)$	
	(lower bound)	[43]	$\Omega(\sqrt{\log d})$	for embedding into ℓ_1
		This paper	$O(\alpha\log\log n)$	space $n^{2+\varepsilon}\cdot 2^{d/\alpha}$ and query time is $d^{O(1)}n^{\varepsilon}$

[a] Unless mentioned otherwise, query time is $d\log^{O(1)} n$ and space is $(dn)^{O(1)}$.

Table 1: Our results for the Ulam and EMD metrics, compared with previous bounds.

over the $O(\log d)$ approximation that follows immediately from the ℓ_1-embedding of [16]. It is known that $O(1)$-approximation requires sketch size $\Omega(\log\log d)$ and, in fact, a logarithmic lower bound is quite plausible [8].

Our sketch is in fact computable in the data stream model, thus answering a question from [2, Section 6] on computing Ulam distance in a stream. Specifically, we can compute the sketch of a string P even if we have a sequential access to elements $P[1], P[2], \ldots$, using a total of $\mathrm{polylog}(d)$ space.

Sublinear Distance Estimation. The third problem is perhaps the most natural: it is that of computing the distance between two given points x and y, potentially up to a ρ-approximation. The goal here is to obtain the best possible running time, as a function of the size of the point representation (in our case d). Where possible, the best-case scenario would be a sublinear time algorithm.

We present the first algorithm for Ulam's distance that achieves sublinear distance estimation within a constant factor in sublinear time. The algorithm's running time for two strings P, Q is $\tilde{O}(d/\sqrt{\mathrm{ed}(P,Q)})$.[1] Our algorithm's running time is optimal in the two extremes, namely, when $\mathrm{ed}(P,Q) = O(1)$ and when $\mathrm{ed}(P,Q) = \Omega(d)$, since a query complexity lower bound of $\Omega(\sqrt{\mathrm{ed}(P,Q)} + d/\mathrm{ed}(P,Q))$ can be obtained by arguments similar to those in [14, Section 4]. Our approximation improves over the d^{ϵ} given in [14] for edit distance in general strings.

We remark that our sublinear time algorithm for Ulam metric has already found another application — it was recently used in [9] to obtain a sublinear time algorithm for computing edit distance between smoothed 0-1 strings. The main idea there is to compute on the fly a reduction between the two problems that maintains "locality of reference".

[1] Following standard convention, we use $\tilde{O}(f(n))$ to mean $O(f(n)\cdot \log^{O(1)} f(n))$.

NNS for EMD metric. For EMD over $[d]^2$, our techniques, together with an embedding from [29], yield NNS with $O(\frac{\alpha}{\varepsilon}\log\log n)$ approximation, $d^{O(1)}n^{\varepsilon}$ query time and $n^{2+\varepsilon}\cdot 2^{d^{1/\alpha}}$ space for any desired $\alpha = \alpha(d,n) > 1$ and $\varepsilon > 0$. This improves upon the $O(\log d)$ approximation of [15, 32, 43], albeit at the cost of a much higher space. Our approximation also beats the $\Omega(\sqrt{\log d})$ non-embeddability lower bound into ℓ_1 as long as $n \ll \exp(\exp(\sqrt{\log d}))$ [43]. We defer details to the full version of the paper.

1.3 Overview of Techniques

Our technical contributions are twofold. The first part shows an embedding of the Ulam metric into a product space. The second part gives efficient algorithms for the resulting product metrics.

Embedding Ulam into product spaces. Let Ulam_d denote the Ulam metric over strings of length d over alphabet Σ (we omit Σ for simplicity). Our first contribution is the following embedding.

THEOREM 1.1. *There exists a constant distortion embedding* $\varphi : \mathrm{Ulam}_d \mapsto \bigoplus_{(\ell_2)^2}^{d} \bigoplus_{\ell_\infty}^{O(\log d)} \ell_1^{2d}$.

Our new embedding of Ulam metric is based on a new estimate of Ulam's distance. It is inspired by previous work on testing and estimating the *distance to monotonicity/sortedness* [21, 1, 23], but unlike the previous estimates which are asymmetric in the two strings, our new estimate is entirely symmetric in the two strings. Our estimate uses primitives such as "count", "there exists", and "majority", but we design the estimate such that it can be transformed into a distance function, in fact a norm, defined via iterated product spaces. The resulting embedding turns out to be very simple to compute, mapping a string into a carefully chosen collection of incidence vectors of its substrings.

Algorithms for product spaces. Our second contribution is designing efficient algorithms for iterated product spaces. For example, the following theorem, together with Theorem 1.1, already gives an NNS scheme for Ulam with $O(\log\log n)$ approximation (the improved $O(\log\log d)$ approximation is obtained by extending the two theorems).

THEOREM 1.2. *For every $k, l, m > 1$ and $\epsilon > 0$, the metric $\bigoplus_{(\ell_2)^2}^k \bigoplus_{\ell_\infty}^l \ell_1^m$ admits an NNS scheme achieving approximation $O(\epsilon^{-3}\log\log n)$, query time $(klm)^{O(1)}n^\epsilon$, and space $(klm)^{O(1)}n^{2+\epsilon}$.*

All our NNS algorithms build on a new NNS scheme that we design for a sum-product metric $\bigoplus_{\ell_1}\mathcal{M}$. This latter scheme uses a technique, which we call *black-box* Locality Sensitive Hashing (LSH). LSH-type techniques have been used before for NNS under simple metrics like ℓ_1 and ℓ_2, and are based on probabilistic partitionings of the corresponding space. Naturally, for a metric like $\bigoplus_{\ell_1}\mathcal{M}$, we cannot hope to do a similar partitioning of the space since we do not have any information about the metric $\mathcal{M}$. However, we show the space can be partitioned in a black-box manner so as to effectively reduce NNS for $\bigoplus_{\ell_1}\mathcal{M}$ to NNS for max-product $\bigoplus_{\ell_\infty}\mathcal{M}$, with a mild increase in parameters. (See Theorem 3.2 for more details.) For the latter max-product $\bigoplus_{\ell_\infty}\mathcal{M}$, we can use the algorithms of [25, 27]. We note that a related idea was also present in [28]. However, the algorithm of [28] had much larger (superlogarithmic) approximation factor, which makes it inapplicable to the scenarios we consider in this paper.

Our sketching algorithm uses two tools: sub-sampling (i.e., projecting a vector on a random subset of coordinates) and sketching of *heavy hitters* [18] (which enable the recovery of coordinates on which the two sketched vectors differ considerably). This idea is somewhat related to the L_k norm estimation algorithm of [33], although the technical development is very different here. We do not provide a sketch of the space $\bigoplus_{(\ell_2)^2}^k \bigoplus_{\ell_\infty}^l \ell_1^m$ in its full generality, and it is in fact plausible that short sketches for this norm might not exist. Instead, we make use of additional properties of our embedding's images. Finally, to obtain a data stream algorithm for computing the sketch, we employ the *block* heavy hitters algorithm of [5], as well as a technique of an attenuated window in the stream.

1.4 Related Work

Product spaces. Product spaces were studied in the context of developing NNS for other hard metrics in [27, 28]. An algorithm for NNS under the max-product $\bigoplus_{\ell_\infty}^k\mathcal{M}$ is designed in [27], achieving $O(c\log\log n)$ approximation using polynomial space and sublinear query time, under the assumption that $\mathcal{M}$ itself has an NNS scheme achieving c-approximation with polynomial space and sublinear query time. Although [28] gave two algorithms for NNS under the sum-product $\bigoplus_{\ell_1}^k\mathcal{M}$, they are much less satisfying, since one requires very large storage and the other obtains a rather large approximation. Our NNS algorithm significantly improves the NNS for sum-products from [28], achieving performance comparable to that of max-products.

We note that the algorithms from [28] were used to design algorithms for the NNS under the edit distance. However, they did not provide any embedding of the edit distance into a simpler space, and thus do not fall under our approach of identifying richer host spaces. There has also been work on *streaming* product metrics such as $\bigoplus_{\ell_p}\ell_q$ (see, [19, 34]). Furthermore, product spaces, even iterated ones, are examined quite frequently in the study of the geometry of Banach spaces, see e.g. [35, Chapter 1].

Nearest Neighbor Search. For edit distance in general strings, the two known NNS schemes with strongly sublinear query time achieve a constant factor approximation using n^{d^ε} storage for every fixed $\varepsilon > 0$ [28], or $2^{O(\sqrt{\log d \log\log d})}$ approximation using $(dn)^{O(1)}$ storage [44]. The latter result is obtained by embedding the corresponding metric into ℓ_1.

For EMD, the only known NNS scheme achieves $O(\log d)$ approximation with polynomial storage, and is also obtained via embedding into ℓ_1 [15, 32, 43].

Sketching Algorithms. Most known algorithms for NNS under edit distance, Ulam, and EMD actually go through ℓ_1 embeddings, as we just mentioned. Thus, they have $O(1)$ sketch size. The only known lower bound on sketching complexity applies to both edit distance on 0-1 strings and to Ulam metric [8], showing that the sketch size must be $\Omega(\log\frac{\log d}{\rho\log\rho})$ for approximation ρ.

Distance Estimation. Sublinear time (heuristic) algorithms are often used as a filtering step in sequence alignment tools to weed out sure non-matches (cf. [17]). Clearly, sublinear time is not always possible, but it is concievable when the edit distance is relatively high. The only previously known sublinear time algorithm, due to [14], works for general strings and can distinguish, with high probability whether $\mathrm{ed}(x,y) \le d^{1-\epsilon}$ or $\mathrm{ed}(x,y) \ge \Omega(d)$, in time $\tilde{O}(d^{\max\{1/2-\epsilon/2, 1-2\epsilon\}})$.

1.5 Preliminaries

Let Σ be the alphabet. For $x \in \Sigma^d$, we use the notation x_i or $x[i]$ to refer to the i^{th} position in x.

For $P, Q \in \mathrm{Ulam}_d$, we let $\underline{\mathrm{ed}}(P,Q)$ denote the minimum number of deletions from P to obtain a subsequence of Q. Note that $\underline{\mathrm{ed}}(Q,P) = \underline{\mathrm{ed}}(P,Q)$ and $\underline{\mathrm{ed}}(P,Q) \le \mathrm{ed}(P,Q) \le 2\,\underline{\mathrm{ed}}(P,Q)$.

To simplify the presentation in the rest of the paper, we will assume that $\Sigma = [d]$ since we can reduce the more general case to it. For example, here is one simple reduction from Ulam on alphabet Σ with $|\Sigma| > d$ to Ulam metric over strings of length $|\Sigma|$ and alphabet Σ: for given $x \in \Sigma^d$, construct $\tilde{x} \in \Sigma^{|\Sigma|}$ by appending all the alphabet symbols that are missing from x in the increasing order. Then, for any $x, y \in \Sigma^d$, $\mathrm{ed}(\tilde{x}, \tilde{y})$ is within a factor 3 of $\mathrm{ed}(x, y)$. Furthermore, if $|\Sigma| \gg d^{O(1)}$, one can use standard hashing to reduce the alphabet to a polynomial in d size.

We will use the notation $\bigoplus_{\ell_p} \mathcal{M}$ for product metric, as defined in the Introduction. Abusing terminology, we shall continue to call $\bigoplus_{\ell_p} \mathcal{M}$ a metric even when it is not guaranteed to satisfy the triangle inequality, for example, in the case of $\bigoplus_{(\ell_2)^2} \mathcal{M}$. For the distance function in an arbitrary iterated product space, we use the subscript to identify which operations are made in which order. For example $d_{\text{NEG},\infty,1}$ is the distance function of the space $\bigoplus_{(\ell_2)^2} \bigoplus_{\ell_\infty} \ell_1$.

2 Embedding the Ulam Metric into Product Spaces

We now present our embeddings of Ulam into product spaces. Our first and main embedding is a more complete statement of Theorem 1.1 and embeds Ulam in $\bigoplus_{(\ell_2)^2}^{d} \bigoplus_{\ell_\infty}^{O(\log d)} \ell_1^{2d}$.

Theorem 1.1 (Restated) *For every $d \geq 1$, there exists an embedding $\varphi : \mathrm{Ulam}_d \mapsto \bigoplus_{(\ell_2)^2}^{d} \bigoplus_{\ell_\infty}^{O(\log d)} \ell_1^{2d}$ such that for all $x, y \in \mathrm{Ulam}_d$:*

$$\mathrm{ed}(x, y) \leq d_{\text{NEG},\infty,1}(\varphi(x), \varphi(y)) \leq O(1) \cdot \mathrm{ed}(x, y).$$

When φ is viewed as an embedding into $\ell_1^{O(d^2 \log d)}$, it has distortion $O(\log^2 d)$. The image $\varphi(x)$ of an input string x can be computed in time $O(d^2 \log d)$.

From this embedding, we also derive a second embedding. The second embedding has the advantage of a somewhat simpler host space, $\bigoplus_{(\ell_2)^2} \ell_\infty$, however, it handles only one scale of distances and has high dimension (in our applications). We use both embeddings to further improve the approximation of NNS from $O(\log \log n)$ (given by Theorem 1.1 alone), to $O(\log \log d)$.

LEMMA 2.1. *For every $1 \leq R, \alpha \leq d$ there is a randomized map $\hat{\varphi} : \mathrm{Ulam}_d \to \bigoplus_{(\ell_2)^2}^{d^3} \ell_\infty^m$ with $m = d^{O(\alpha)}$, such that for every $x, y \in \mathrm{Ulam}_d$, with probability at least $1 - e^{-\Omega(d^2)}$ we have: If $\mathrm{ed}(x, y) \geq R$ then $d_{\text{NEG},\infty}(\hat{\varphi}(x), \hat{\varphi}(y)) \geq \Omega(R)$, and if $\mathrm{ed}(x, y) \leq R/\alpha$ then $d_{\text{NEG},\infty}(\hat{\varphi}(x), \hat{\varphi}(y)) \leq O(R/\alpha)$.*

We prove Theorem 1.1 below and defer the proof of the Lemma 2.1 to the full version of this article. We start by presenting the construction of the embedding φ from Theorem 1.1.

Construction of φ. We use the following notation. For $P \in \mathrm{Ulam}_d$, we assume by convention that in positions $j = 0, -1, \ldots, -d+1$ we have $P[j] = j$ and set the *extended* alphabet to be $\bar{\Sigma} = \{-d+1, \ldots, d\}$. For $a \in [d]$ and $k \in [d]$, let P_{ak} be a set containing the k symbols that appear in the k positions immediately before symbol a in P, i.e. $P_{ak} = \{P[P^{-1}[a] - k], \ldots, P[P^{-1}[a] - 1]\}$.

We proceed in three steps. First, for a symbol $a \in [d]$ and integer $k \in [d]$, we define $\varphi_{ak} : \mathrm{Ulam}_d \mapsto \ell_1^{2d}$ by setting $\varphi_{ak}(P)$ to be the $0/1$ incidence vector of P_{ak} scaled by $1/2k$. Thus, $\varphi_{ak}(P) \in \{0, \frac{1}{2k}\}^{\bar{\Sigma}}$ and has exactly k nonzero entries. Distances in this host space are computed using the ℓ_1-norm, namely $\|\varphi_{ak}(P) - \varphi_{ak}(Q)\|_1$. Second, for every $a \in \Sigma$, define $\varphi_a : \mathrm{Ulam}_d \mapsto \bigoplus_{\ell_\infty}^{O(\log d)} \ell_1^{2d}$ to be the direct sum $\varphi_a(P) = \oplus_{k \in K} \varphi_{ak}(P)$, where $K = \{\lceil (1+\gamma)^i \rceil : i = 0, 1, \ldots, \lceil \log_{1+\gamma} d \rceil\}$ ranges over all powers of $1+\gamma$ in $[d]$ where we set $\gamma = 1/4$.[2] Distances in this product space are computed using an ℓ_∞-norm, namely $d_{\infty,1}(\varphi_a(P), \varphi_a(Q)) = \max_{k \in K} \|\varphi_{ak}(P) - \varphi_{ak}(Q)\|_1$. Third, define $\varphi : \mathrm{Ulam}_d \mapsto \bigoplus_{(\ell_2)^2}^{d} \bigoplus_{\ell_\infty}^{O(\log d)} \ell_1^{2d}$ by the direct sum $\varphi(P) = \oplus_{a \in [d]} \varphi_a(P)$. Distances in this host space are computed using squared-ℓ_2, i.e. $d_{\text{NEG},\infty,1}(\varphi(P), \varphi(Q)) = \sum_{a \in [d]} \left(d_{\infty,1}(\varphi_a(P), \varphi_a(Q))\right)^2$.

An estimate of the Ulam distance. The following lemma is key to the proof of Theorem 1.1 and provides an estimate on the Ulam distance between two permutations. It is inspired by, and technically builds upon, [21, 1, 23], which gave estimates to the distance from P to a fixed permutation, say the identity (hence called distance to monotonicity). Relabeling of the symbols can clearly be used to apply these previous estimates to two arbitrary permutations P and Q; however, it requires an explicit description of Q^{-1}, which is inefficient or just impossible, in our intended applications.[3] Thus, the main advantage of our estimate is that it is the first one which is *efficient* for two *arbitrary* permutations. In the sequel, we use $A \triangle B$ to denote the symmetric difference between two sets A, B.

LEMMA 2.2. *Fix $P, Q \in \mathrm{Ulam}_d$, and let $0 < \delta \leq 1/2$. Let T_δ be the set containing all symbols $a \in \Sigma$ for which there exists $k \in [d]$ such that the symmetric difference $|P_{ak} \triangle Q_{ak}| > 2\delta k$. Then*

$$\frac{1}{2}\,\underline{\mathrm{ed}}(P, Q) \leq |T_\delta| \leq \frac{4}{\delta} \cdot \underline{\mathrm{ed}}(P, Q). \tag{2.1}$$

In the case of two permutations P and Q, there is a crucial (albeit technical) difference between our estimate and the previous ones, including [1, 23]. The core of all such estimates is a certain counting. In our case, for a given symbol a and integer k, it is $|P_{ak} \triangle Q_{ak}|$ (the symmetric difference between the k symbols appearing immediately before a in P and similarly in Q); and it is well-known that symmetric difference can be expressed as the ℓ_1 difference between the respective incidence vectors. In contrast, previous estimates

[2] To simplify the exposition, we shall ignore rounding issues and the fact that the largest value in K should be capped by d.

[3] We seek embeddings that are oblivious, i.e. the image of P has to be determined independently of Q. In NNS algorithms, the data string P are preprocessed without knowing the query Q. Sublinear algorithms cannot afford to compute Q^{-1} explicitly, as it would take linear time.

ask how many of the k symbols appearing immediately before a in P (i.e. the set P_{ak}), appear in Q *after* a. Such a set-intersection formulation does not lend itself to embeddings.[4] In this sense, our estimate is symmetric with respect to P and Q, while previous ones are not. Nevertheless, our proof relies on the technical analysis of [1, 23], but in a rather nontrivial way. In particular, we restore symmetry between the two permutations by applying the known bounds twice, once from P towards Q and once from Q towards P. The full proof of Lemma 2.2 follows. It is more convenient for us to use the notation and analysis from [23] (rather than [1]).

Proof. [Proof of Lemma 2.2] Fix $P, Q \in \mathrm{Ulam}_d$ and $0 < \delta \leq 1/2$. We say that two distinct symbols $a, b \in \Sigma$ are inverted in P vs. in Q if these symbols do not appear in the same order in P and in Q, i.e. if $(P^{-1}[a]-P^{-1}[b])(Q^{-1}[a]-Q^{-1}[b]) < 0$. We say that a pair of indexes i, j in P is inverted if the respective symbols $P[i], P[j]$ are inverted. Define a set R^P_δ containing all indexes $i \in [d]$ for which there is $j < i$ such that for more than δ-fraction of indexes $j' \in [j, i-1]$ the pair of indexes i, j' is inverted in P. We know from [23, Lemma 3.1] that

$$\underline{\mathrm{ed}}(P,Q) \leq 2|R^P_{1/2}|. \tag{2.2}$$

(It is assumed therein that Q is the identity permutation; the bound above may seem more general but it follows immediately by relabeling symbols.) We claim that $R^P_{1/2} \subseteq R^P_\delta \subseteq T_\delta$. Indeed, whenever $a \in R^P_{1/2}$, there is $j < i$ such that more than $1/2 \geq \delta$ of the indexes $j' \in [j, i-1]$ are inverted with respect to i in P, and in particular $P[j'] \in P_{a,i-j} \setminus Q_{a,i-j}$. Since $|P_{a,i-j}| = |Q_{a,i-j}| = i-j$, it follows that $|P_{a,i-j} \triangle Q_{a,i-j}| = 2|P_{a,i-j} \setminus Q_{a,i-j}| > 2\delta(i-j)$, and thus $a \in T_\delta$, proving the claim. Using the claim and (2.2), we have $\underline{\mathrm{ed}}(P,Q) \leq 2|T_\delta|$, which proves the first inequality in (2.1).

We proceed to proving the second inequality in (2.1). Fix an optimal alignment between P and Q, namely a subset $D \subseteq \Sigma$, $|D| = \underline{\mathrm{ed}}(P,Q)$ such that deleting the symbols of D from P and from Q yields identical strings. Let $D^P = \{i \in [d] : P[i] \in D\}$ denote the indexes of D in P, and define D^Q similarly for Q. Define a set S^P_δ containing all indexes $i \in [d]$ for which there is $j < i$ such that more than δ-fraction of indexes $j' \in [j, i-1]$ belong to D^P. Let S^Q_δ be defined similarly for Q. We then know from [23, Lemma 3.2],[5] that for all $0 < \delta' \leq 1/2$,

$$|R^P_{\delta'} \setminus D^P| \leq |S^P_{\delta'}| \leq (1-\delta')/\delta' \cdot |D^P|,$$

and, in fact, that $R^P_{\delta'} \setminus D^P \subseteq S^P_{\delta'}$. Therefore we deduce that

$$|R^P_{\delta'} \cup S^P_{\delta'}| \leq |D^P \cup S^P_{\delta'}| \leq \tfrac{1}{\delta'} \cdot |D^P|, \tag{2.3}$$

and similarly for Q.

We next show that

$$|T_\delta| \leq |R^P_{\delta/2} \cup S^P_{\delta/2}| + |R^Q_{\delta/2} \cup S^Q_{\delta/2}|. \tag{2.4}$$

Indeed, consider $a \in T_\delta$ and let $k \in [d]$ be its witness, namely $|P_{ak} \triangle Q_{ak}| > 2\delta k$. The case where $P^{-1}[a] \in R^P_{\delta/2} \cup S^P_{\delta/2}$ can be paid for using the term $|R^P_{\delta/2} \cup S^P_{\delta/2}|$. The case where $Q^{-1}[a] \in R^Q_{\delta/2} \cup S^Q_{\delta/2}$ can be paid for using the term $|R^Q_{\delta/2} \cup S^Q_{\delta/2}|$. We now claim that these are the only two possible cases, i.e. not being in either of the two cases implies a contradiction. Indeed, if $a \in T_\delta$ and $P^{-1}[a] \notin R^P_{\delta/2} \cup S^P_{\delta/2}$, then there must be at least one symbol $b' \in \bar{\Sigma}$ such that **(a)** $b' \in P_{ak} \setminus Q_{ak}$; **(b)** b' is not inverted wrt to a; and **(c)** b' is not in D. Using **(a)** and **(b)** we have that **(d)** b' appears in Q more than k positions before a (i.e. its index in Q is smaller than $Q^{-1}[a] - k$). Since also $Q^{-1}[a] \notin R^Q_{\delta/2} \cup S^Q_{\delta/2}$, we similarly obtain a symbol $b'' \in \bar{\Sigma}$ such that **(a')** $b'' \in Q_{ak}$; **(c')** b'' is not in D; and **(d')** b'' appears in P more than k positions before a. We obtain from **(a)** and **(d')** that b' appears after b'' in P, and from **(a')** and **(d)** that b'' appears after b' in Q. Thus, the symbols b', b'' are inverted, and at least one of them must belong to D, contradicting **(c)** and **(c')**. This proves the claim and (2.4).

Finally, using (2.3), (2.4), and the fact that $|D^P| = |D^Q| = \underline{\mathrm{ed}}(P,Q)$, we conclude $|T_\delta| \leq \frac{2}{\delta}|D^P| + \frac{2}{\delta}|D^Q| = \frac{4}{\delta}\,\underline{\mathrm{ed}}(P,Q)$, which proves the second inequality in (2.1), and completes the proof of Lemma 2.2. ■

We can now complete the proof of Theorem 1.1 using Lemma 2.2. We need to bound the distortion of φ when viewed as an embedding into $\bigoplus_{(\ell_2)^2} \bigoplus_{\ell_\infty} \ell_1$. In a nutshell, the distortion of the embedding roughly corresponds to $\sum_{\delta=2^{-j}} \delta^2 |T_\delta| / \mathrm{ed}(P,Q) \leq O(\sum_{\delta=2^{-j}} \delta) \leq O(1)$, where the squared term comes from the outer ℓ_2-squared product, and would not work if instead we were to use ℓ_1 as the outer product.

Proof. [Proof of Theorem 1.1] Fix two distinct permutations $P, Q \in \mathrm{Ulam}_d$. By definition, for all $a \in \Sigma$ and $k \in [d]$ we have $\|\varphi_{ak}(P) - \varphi_{ak}(Q)\|_1 = \frac{1}{2k}|P_{ak} \triangle Q_{ak}|$.

We first bound $d_{\mathrm{NEG},\infty,1}(\varphi(P), \varphi(Q))$ from below. By Lemma 2.2 (and its notation) for $\delta = 1/2$, we know that $|T_\delta| \geq \frac{1}{2}\,\underline{\mathrm{ed}}(P,Q)$. Now fix $a \in T_\delta$. Then there exists $k \in [d]$ such that $|P_{ak} \triangle Q_{ak}| > 2\delta k$, and rounding this k upwards to the next power of $1+\gamma$, we obtain $k' \in K$ such that $\|\varphi_{ak'}(P) - \varphi_{ak'}(Q)\|_1 = \frac{1}{2k'}|P_{ak'} \triangle Q_{ak'}| \geq \frac{1}{2k'}(2\delta k - 2\gamma k) = \frac{\delta-\gamma}{1+\gamma} = \frac{1}{5}$. (We remind that the rounding issues we neglected would lead to slightly worse constants.) Thus, for each $a \in T_\delta$ we have $d_{\infty,1}(\varphi_a(P), \varphi_a(Q)) \geq 1/5$, and thus

$$d_{\mathrm{NEG},\infty,1}(\varphi(P), \varphi(Q)) \geq \sum_{a \in T_\delta} (\tfrac{1}{5})^2 \geq \tfrac{\underline{\mathrm{ed}}(P,Q)/2}{25} = \tfrac{\underline{\mathrm{ed}}(P,Q)}{50}.$$

[4] Similarly, our estimate lends itself to sublinear sampling, while under the previous estimates, it seems to require access to Q^{-1}.

[5] As pointed out in [23], a similar upper bound, up to constant factors, is implied by results of [1, Lemma 2.3].

To bound $d_{\text{NEG},\infty,1}(\varphi(P),\varphi(Q))$ from above, we relax the range $k \in K$ into $k \in [d]$, and break the contribution arising from different $a \in \Sigma$ into buckets of the form $[2^{-j}, 2^{-j+1}]$.

$$d_{\text{NEG},\infty,1}(\varphi(P),\varphi(Q)) = \sum_{a\in\Sigma} \Big(d_{\infty,1}(\varphi_a(P),\varphi_a(Q))\Big)^2 \le$$

$$\le \sum_{a\in\Sigma} \max_{k\in[d]} \|\varphi_{ak}(P) - \varphi_{ak}(Q)\|_1^2 =$$

$$= \sum_{a\in\Sigma} \max_{k\in[d]} [\tfrac{1}{2k}|P_{ak} \triangle Q_{ak}|]^2 \le 1 + \sum_{j=1}^{\log d} (2^{-j+1})^2 \cdot |T_{2^{-j}}|.$$

By Lemma 2.2, we have $|T_{2^{-j}}| \le 2^{j+2} \cdot \underline{\text{ed}}(P,Q)$, and therefore

$$d_{\text{NEG},\infty,1}(\varphi(P),\varphi(Q))$$
$$\le 1 + \sum_{j=1}^{\log d} 2^{-j+4} \cdot \underline{\text{ed}}(P,Q) \le 17 \cdot \underline{\text{ed}}(P,Q).$$

The second part of the theorem results from a similar computation on $\|\varphi(P) - \varphi(Q)\|_1$. ■

3 New Algorithms for Product Metrics

We present new algorithms for three applications: nearest neighbor search (NNS), sketching, and sublinear distance estimation. We first develop algorithms for NNS over general product spaces, and subsequently obtain NNS for Ulam, relying on our Ulam embeddings. The other two applications, sketching (including streamable sketching) and sublinear distance estimation rely on the Ulam embedding from Theorem 1.1, but apply to Ulam only.

3.1 NNS for Product Spaces We now design new NNS schemes for product spaces. The main ingredient to all NNS algorithms is the theorem below that reduces an ℓ_1-product metric to an ℓ_∞-product metric. In the sequel, we also use the terms sum-product and max-product, respectively. Combining it with the NNS for max-product metrics from [25, 27], we obtain a general composition principle that is useful for constructing NNS scheme for iterated products with respect to ℓ_1, ℓ_2-squared, and ℓ_∞. Throughout, we let n denote the number of points in the NNS dataset, and will not worry about preprocessing times since it is the same as the space bound in all our algorithms.

We note that our result does not require the triangle inequality and is thus applicable also to ℓ_2-squared. On the other hand, we require the following quite natural property of a metric space.

DEFINITION 3.1. *Let $(\mathcal{M}, d_{\mathcal{M}})$ be a metric space. A map $\sigma : \mathcal{M} \to \mathcal{M}$ is called an* α-dilation*, for $\alpha > 0$, if for all $x, y \in \mathcal{M}$ we have $d_{\mathcal{M}}(\sigma(x),\sigma(y)) = \alpha \cdot d_{\mathcal{M}}(x,y)$. The metric is called* scalable *if for every $\alpha > 0$ it has an α-dilation σ_α. To simplify notation, we write $\alpha \cdot x$ for $\sigma_\alpha(x)$.*

THEOREM 3.2. (NNS FOR SUM-PRODUCT) *Let $(\mathcal{M}, d_{\mathcal{M}})$ be a scalable metric space and let $k \ge 1$. Suppose there is an NNS scheme for the max-product $\bigoplus_{\ell_\infty}^k \mathcal{M}$ with approximation c, query time $Q(n)$, and space $S(n)$. Then for every $\epsilon > 0$ there is an NNS scheme for the sum-product $\bigoplus_{\ell_1}^k \mathcal{M}$ with approximation $\tilde{c} = O(\frac{1}{\epsilon}c)$, query time $kn^\epsilon \cdot Q(n)$, and space $kn^\epsilon \cdot S(n)$.*

An immediate corollary of this theorem is the Theorem 1.2. As previously mentioned, this in turn implies an immediate NNS for Ulam with $O(\log\log n)$ approximation. Later, we further improve the approximation to $O(\log\log d)$, with another application of Theorem 3.2 together with the second Ulam embedding from Lemma 2.1 (see details in Corollary 3.4).

Proof. [Proof of Theorem 3.2] We design an algorithm for the decision version of approximate NNS problem, namely $\tilde{c}$-approximate *near*-neighbor. In this decision problem, given a dataset D and a radius R, we construct a data structure that, given a query point q, if there is a point $p \in D$ with $d(p,q) \le R$ then the data structure reports, with probability at least $1/2$, a point $p' \in D$ such that $d(p',q) \le \tilde{c} \cdot R$. An algorithm for this problem implies an algorithm for approximate NNS by the results of [31, 24].

The main idea is to design a generalization of Locality Sensitive Hashing (LSH). Previously, LSH has been used to design NNS under simple metrics like ℓ_1 and ℓ_2. In a nutshell, LSH is a (non-adaptive) hashing scheme that probabilistically partitions the entire space into buckets such that a pair of "close" points (distance $\le R$) have higher probability of collision (i.e., falling into the same bucket) than a pair of "far" points (distance $\ge \tilde{c}R$). The NNS algorithm then hashes all n data points according to this partition and builds a hash table. Upon recieving a query q, the algorithm computes the hash of q and linearly scans the data points that fall in the same bucket and reporting those that are indeed close to q. To guarantee a constant success probability, the algorithm needs to construct some number $L = L(n, \tilde{c})$ of such hash tables.

Ideally, we would like to be able to similarly partition the space $\bigoplus_{\ell_1} \mathcal{M}$, however we cannot do this since we have no control over $\mathcal{M}$. Nonetheless, we manage to do so in a black-box manner, as will be seen later, replacing a hash table structure by a nearest neighbor data structure for $\bigoplus_{\ell_\infty} \mathcal{M}$. Our algorithm may be viewed as a generalization of the LSH scheme for ℓ_1 in [6]. We now describe our algorithm in detail.

Preprocessing stage. Fix a threshold radius $R > 0$ and let $w = R\log n$. For integers L, t to be chosen later, construct L different max-product data structures (these correspond to the L hash tables of an LSH scheme). For each $i \in [L]$, construct one max-product data structure M_i, as follows. For $u \in [t]$, pick uniformly at random reals $s^i_{1,u}, s^i_{2,u}, \dots s^i_{k,u} \in [0, w]$. Then, for $j \in [k]$, let $s^i_j = \min_{u\in[t]}\{s_{j,u}\}$. From the dataset D, construct the dataset $\tilde{D}_i$ containing all $\tilde{x}$ such that $\tilde{x}$ is obtained from $x \in D$ by scaling each coordinate $j \in [k]$ in the product by $1/s_j^i$. In other words, if $x_j \in \mathcal{M}$ is the j^{th} coordinate of $x \in D \subseteq \mathcal{M}^k$, then

$$\tilde{D}_i = \left\{ \tilde{x} = \left(x_1/s_1^i, x_2/s_2^i, \ldots x_k/s_k^i \right) \;\middle|\; x \in D \right\}.$$

Then M_i is simply a near-neighbor data structure for $\bigoplus_{\ell_\infty}^k \mathcal{M}$ metric with $R' = 1$ constructed on the dataset $\tilde{D}_i$.

Query stage. Given a query point $q \in \mathcal{M}^k$, iteratively go over M_i for all $i \in [L]$. For each M_i, compute $\tilde{q} = (q_1/s_1^i, q_2/s_2^i, \ldots, q_k/s_k^i)$ and query the point $\tilde{q}$ in M_i. For each returned point p, compute the true distance from q to p and report the point p if $d_{1,\mathcal{M}}(p,q) \le \tilde{c}R$, where $\tilde{c} = O(\frac{1}{\epsilon}c)$.

Correctness and running time. Fix one NNS data structure M_i. Consider the reals $\{s_{j,u}^i\}_{j\in[k],u\in[t]}$ used to construct M_i. Now, for $0 \le \alpha \le c$, let us say that two points $p, q \in \bigoplus_{\ell_1}^k \mathcal{M}$ *α-collide* if $d_{\mathcal{M}}(p_j/s_{j,u}, q_j/s_{j,u}) \le \alpha$ for all $j \in [k]$ (this corresponds to collision in an LSH bucket but in a "fuzzy" sense, as determined by α). Observe that if there exists a point $p \in D$ that 1-collides with q for all $u \in [t]$, then the query to M_i will return a point p' that c-collides with q for all $u \in [t]$ (since the NNS structure M_i attains approximation factor c).

It remains to compute the probabilities of α-collision of points p and q. Following the calculations done in [4, Lemma 4.1.1], we have that: (1) if $d_{1,\mathcal{M}}(p,q) \le R$ then the probability of 1-collision of p and q is $\pi_1 \ge 1 - 1/\log n$; and (2) if $d_{1,\mathcal{M}}(p,q) \ge \tilde{c}R$ then the probability of c-collision is $\pi_2 \le 1 - \frac{\tilde{c}/c}{\log n + \tilde{c}/c}$.

We can now choose $L = n^\rho$, where $\rho = \frac{\log 1/\pi_1}{\log 1/\pi_2}$, and $t = \frac{\log n}{\log 1/\pi_2}$. Notice that we can compute the probability of success using standard LSH analysis similar to [31]. It follows that our algorithm will return a point $p' \in D$ at distance at most $\tilde{c}R$ with probability bounded away from zero, whenever there is a point $p \in D$ at distance $\le R$ from q. As usual, the probability can be amplified to any desired value. Note that we have used $L = n^\rho = O(n^{c/\tilde{c}}) = n^{O(\epsilon)}$.

Since we can implement each M_i with query time $Q(n)$ and space $S(n)$, the final data structure has query time $O(Lk \log^2 n) \cdot Q(n) = kn^{O(\epsilon)}Q(n)$, and similarly space $kn^{O(\epsilon)}S(n)$. Scaling ϵ accordingly concludes the proof of the theorem. ■

We now show how the above theorem implies Theorem 1.2, namely an NNS for $\bigoplus_{(\ell_2)^2} \bigoplus_{\ell_\infty} \ell_1$ metric.

Proof. [Proof of Theorem 1.2] Notice that the metric $\bigoplus_{(\ell_2)^2}^k \bigoplus_{\ell_\infty}^l \ell_1^m$ is the same as the metric $\bigoplus_{\ell_1}^k \bigoplus_{\ell_\infty}^l (\ell_1^m)^2$, where $(\ell_1^m)^2$ is the square of the distances in ℓ_1^m, i.e. it is $\mathbb{R}^m$ equipped with distance $d_{1^2}(x,y) = (\sum_{i=1}^m |x_i - y_i|)^2$. We design an NNS scheme for the metric $\bigoplus_{\ell_1}^k \bigoplus_{\ell_\infty}^l (\ell_1^m)^2$. First, we apply Theorem 3.2 to reduce the problem to designing NNS for $\bigoplus_{\ell_\infty}^k \bigoplus_{\ell_\infty}^l (\ell_1^m)^2 = \bigoplus_{\ell_\infty}^{kl} (\ell_1^m)^2$. For this resulting max-product, we use the following theorems from [25, 27].

THEOREM 3.3. ([27, THEOREM 1], [25]) *Consider metric $(\mathcal{M}, d_{\mathcal{M}})$ with an NNS under $\mathcal{M}$ that has approximation c, query time $Q(n)$, and space $S(n)$.*[6] *Then, for every $\epsilon > 0$, there exists NNS under $\bigoplus_{\ell_\infty}^k \mathcal{M}$ with approximation $O(\epsilon^{-1} \log\log n)$, query time $O((Q(n) + kd)\log n)$, and space $kS(n)n^{1+\epsilon}$, where d is the time to compute distance in $\mathcal{M}$. If $\mathcal{M} = \mathbb{R}$, the approximation becomes $O(\epsilon^{-1} \log\log k)$.*

We note that $(\ell_1^m)^2$ admits NNS with approximation $1/\epsilon$, query time $O(mn^\epsilon)$ and space $O(mn^{1+\epsilon})$ by [31]. Thus we can apply Theorem 3.3 with $\mathcal{M} = (\ell_1^m)^2$, and obtain NNS under the max-product $\bigoplus_{\ell_\infty}^{kl} (\ell_1^m)^2$. The resulting approximation is $O(\frac{1}{\epsilon}) \cdot O(\frac{1}{\epsilon}) \cdot O(\frac{1}{\epsilon} \log\log n)$ coming from Theorem 3.2, the NNS for $(\ell_1)^m$, and Theorem 3.3, respectively. The resulting query time is $(klm)^{O(1)} n^{2\epsilon}$ and space is $(klm)^{O(1)} n^{2+3\epsilon}$. ■

3.2 NNS for Ulam

Using the embeddings of Ulam in Theorem 1.1 and in Lemma 2.1, in conjunction with the NNS data structures from above, we achieve an $O(\log\log d)$ approximation NNS for Ulam as follows.

COROLLARY 3.4. *For every constant $0 < \varepsilon < 1$ there is a randomized NNS algorithm under Ulam_d that achieves approximation $O(\varepsilon^{-3} \log\log d)$, query time $d^{O(1)} n^\varepsilon$, and space $d^{O(1)} n^{2+\varepsilon}$.*

Proof. We employ one of two different data structures, depending on the parameters d and n. If $n \le d^{\log d}$, we apply Theorem 1.1 to embed Ulam into $\bigoplus_{(\ell_2)^2}^d \bigoplus_{\ell_\infty}^{O(\log d)} \ell_1^{2d}$, and use the NNS from Corollary 1.2, achieving approximation $O(\varepsilon^{-3} \log\log n) = O(\varepsilon^{-3} \log\log d)$.

If $n > d^{\log d}$, then we use Lemma 2.1 with $\alpha = O(\log\log d)$, embedding Ulam into $\bigoplus_{(\ell_2)^2}^{d^3} \ell_\infty^m$ for $m = d^{O(\log\log d)}$. Then we can apply Theorem 3.2 together with the NNS for ℓ_∞^k (Theorem 3.3), which achieves $O(\log\log k)$ approximation. Thus, we obtain $O(\varepsilon^{-2} \log\log(d^3 m)) = O(\varepsilon^{-2} \log\log d)$ approximation, with query time $d^{O(\log\log d)} n^\varepsilon \le n^{2\varepsilon}$. ■

3.3 Sketching and Streaming Algorithms

Our general result for sketching of Ulam, in the streaming model, is based on the embedding from Theorem 1.1 and reads as follows:

THEOREM 3.5. *We can compute a randomized sketch $\mathbf{sk} : \mathrm{Ulam}_d \to \{0,1\}^s$ for $s = (\log d)^{O(1)}$ in the streaming model, with $(\log d)^{O(1)}$ time per input character, with the following property. For every $P, Q \in \mathrm{Ulam}_d$, with high probability, given $\mathbf{sk}(P)$ and $\mathbf{sk}(Q)$ only, one can approximate $\mathrm{ed}(P,Q)$ within a constant approximation.*

[6] Strictly speaking, we need to impose a technical condition on the NNS for $\mathcal{M}$, but it is satisfied in all our scenarios; see [27, Section 2] for details.

In this conference version, we give a simpler construction of a polylog-space sketching algorithm Ulam_d, but which is not computable in a stream as is. Nonetheless, this construction already contains some ideas used for obtaining the streamable version of the sketch. Both the analysis of this algorithm, and the streaming version of the sketching result (that uses [5]) are deferred to the full version.

THEOREM 3.6. *There exists a sketching algorithm for* Ulam_d *achieving constant approximation using* $\log^{O(1)} d$ *space.*

Proof. [Proof sketch] Let $P, Q \in \mathrm{Ulam}_d$. We use the notation from Section 2. Notably, let $\varphi, \varphi_a, \varphi_{a,k}$ be as defined in Section 2, and let $\zeta, \zeta_a, \zeta_{a,k}$ be respectively $\varphi(P) - \varphi(Q), \varphi_a(P) - \varphi_a(Q), \varphi_{a,k}(P) - \varphi_{a,k}(Q)$.

We prepare sketches for all possible scales $R = c^i$, $i \in [\log_c d]$, for c a sufficiently large constant, determined later. For each scale we solve the threshold problem: output "far" if $\mathrm{ed}(P, Q) \geq R$ and output "close" if $\mathrm{ed}(P, Q) \leq R/c$, with probability at least $2/3$ (this can be amplified to whp by taking independent sketches). We also assume that $\mathrm{ed}(P, Q) \leq cR$ since the algorithm can enumerate all scales R from the biggest to the smallest stopping as soon as the sketch for the corresponding scale outputs "far".

The main idea is the following. Call $a \in \Sigma$ *expensive character* if $a \in T_{1/2}$, where T_δ is the set of characters z such that $\|\zeta_{z,k}\|_1 > \delta$ from some $k \in K$. In other words, the expensive characters are the ones that contribute a constant fraction to the edit distance (through ζ_a's). To find the expensive characters, we down-sample the characters to some set S such that there are few expensive characters in S. It remains to estimate the number of expensive characters in S.

For an expensive character a, we say it is *expensive at scale* k for some $k \in K$ if $\|\zeta_{a,k}\|_1 > 1/2$. The main observation is that if a is an expensive character at scale k, then $\|\zeta_{a,k}\|_1 \geq \frac{1}{\mathrm{polylog}(d)} \|\{\zeta_{a,k}\}_{a \in S}\|_1$ (this step uses the second part of Theorem 1.1). Now, to find such characters a, we use a sketching algorithm for finding *heavy hitters* under ℓ_1. A more detailed description follows.

DEFINITION 3.7. *Consider a vector* $v \in \mathbb{R}^m$. *Coordinate* i *is called a* ϕ-heavy hitter *if* $|v_i| \geq \phi \|v\|_1$. *The set of* ϕ*-heavy hitters is denoted by* $\mathrm{HH}_1(\phi)$.

We will use the COUNTMIN algorithm, due to Cormode-Muthukrishnan.

LEMMA 3.8. ([18]) *There exists a (randomized) sketching algorithm* $\mathbf{sk}(\cdot)$*, that for every two input vectors* $x, y \in \mathbb{R}^m$ *and parameter* $\phi > 0$*, computes sketches* $\mathbf{sk}(x)$ *and* $\mathbf{sk}(y)$ *of size* $O(\frac{1}{\phi} \log m)$*, such that one can reconstruct from the two sketches a set* $H \subset [d]$*, where, denoting* $\mathrm{HH}_1(\phi)$ *the heavy hitters for the vector* $x - y$*, we have*

$$\Pr[\mathrm{HH}_1(\phi) \subseteq H \subseteq \mathrm{HH}_1(\tfrac{\phi}{2})] \geq 1 - m^{-\Omega(1)}.$$

The algorithm. Fix some R. Remember that we are trying to decide whether $E \geq R$ or $E \leq R/c$, where $E = \underline{\mathrm{ed}}(P, Q)$ (and by assumption $E \leq cR$). Let $S \subset [d]$ be a multi-set of size $|S| = \frac{d}{R} \cdot c_S \log d$, chosen uniformly at random with replacement, where c_S is a large constant.

We want to find, for each $k \in K$, the expensive characters $a \in S$ at scale k. Fix $k \in K$ and consider the multi-set of vectors $\Upsilon_k(P) = \{2k\varphi_{a,k}(P)\}_{a \in S}$, and thus $\Upsilon_k(P) \subseteq \{0,1\}^d$. For $l = \frac{d}{2k} \cdot c_L \log d$, we sub-sample a multi-set $L \subset [d]$ of l coordinates (with replacement). For each $\varphi_{a,k} \in \Upsilon_k(P)$, we let $\varphi^L_{a,k}$ be the restriction of $\varphi_{a,k}$ to the coordinates from L, and let $\Upsilon^L_k(P)$ be the set of $\varphi^L_{a,k}$, where $\varphi_{a,k}$ ranges over $\Upsilon_k(P)$.

Our actual sketch consists of applying the COUNTMIN algorithm (Lemma 3.8) to each set $\Upsilon^L_k(P)$, $k \in K$, to find the positions of all the non-zeros in all the vectors in $\Upsilon^L_k(P)$. For this we view $\Upsilon^L_k(P)$ as a vector of size $|S| \cdot l$, and apply COUNTMIN with $\phi = \Omega(1/\log^4 d)$.

The reconstruction stage for some sketches $\mathbf{sk}(P)$ and $\mathbf{sk}(Q)$ is then the following. Define $\Upsilon_k = \Upsilon_k(P) - \Upsilon_k(Q) = \{2k\zeta_{a,k}(P)\}_{a \in S}$, and similarly with Υ^L_k. Using the linear in φ CountMin sketch $\mathbf{sk}(P) - \mathbf{sk}(Q)$, we find all the non-zeros in Υ^L_k. Once we found all the non-zeros in Υ^L_k, we define the set $X_k \subset S$ of characters which are "near-expensive" according to the subset L: i.e., $a \in X_k$ iff $\|2k\zeta^L_{a,k}\|_1 \geq \frac{1}{3} c_L \log d$. Then, the set of all expensive characters in S is estimated to be $X = \cup_k X_k$. If $|X| \geq \frac{1}{3} \cdot c_S \log d$, then we declare P and Q to be far. Otherwise, P and Q are close. ∎

3.4 Sublinear distance estimation Using the embedding of Ulam from Theorem 1.1, we obtain the following algorithm. We defer the proof of next theorem to the full version of the paper.

THEOREM 3.9. *There is a randomized algorithm that, given* $P, Q \in \mathrm{Ulam}_d$ *estimates* $\mathrm{ed}(P, Q)$ *within a constant factor in time (and query complexity)* $\tilde{O}(d/\sqrt{\mathrm{ed}(P, Q)})$.

4 Conclusions

In this paper we give a constant-distortion embedding of Ulam metric into an (iterated) product space. Such embedding is provably impossible to achieve with simpler host spaces, such as ℓ_1. We further show that this implies improved algorithms for a variety of computational tasks.

The main problem left open by this work is: is it possible to embed the edit distance into a (computationally tractable) iterated product metric with a constant distortion? Such a result could have far-reaching algorithmic implications, both in theory and in practice.

References

[1] N. Ailon, B. Chazelle, S. Comandur, and D. Liu. Estimating the distance to a monotone function. *Random Struct. Algorithms*, 31(3):371–383, 2007.

[2] M. Ajtai, T. S. Jayram, R. Kumar, and D. Sivakumar. Approximate counting of inversions in a data stream. In *Proc. of STOC*, pages 370–379, 2002.

[3] N. Alon, Y. Matias, and M. Szegedy. The space complexity of approximating the frequency moments. *Proc. of STOC*, pages 20–29, 1996.

[4] A. Andoni. Approximate nearest neighbor problem in high dimensions. *M.Eng. Thesis, Massachusetts Institute of Technology*, June 2005.

[5] A. Andoni, K. Do Ba, and P. Indyk. Block heavy hitters. *MIT Technical Report MIT-CSAIL-TR-2008-024*, 2008.

[6] A. Andoni and P. Indyk. Efficient algorithms for substring near neighbor problem. In *Proc. of SODA*, pages 1203–1212, 2006.

[7] A. Andoni, P. Indyk, and R. Krauthgamer. Earth mover distance over high-dimensional spaces. *Proc. of SODA*, pages 343–352, 2008.

[8] A. Andoni and R. Krauthgamer. The computational hardness of estimating edit distance. In *Proc. of FOCS*, pages 724–734, 2007.

[9] A. Andoni and R. Krauthgamer. The smoothed complexity of edit distance. In *Proc. of ICALP*, pages 357–369, 2008.

[10] S. Arora, J. R. Lee, and A. Naor. Euclidean distortion and the sparsest cut. In *Proc. of STOC*, pages 553–562, 2005.

[11] S. Arora, S. Rao, and U. Vazirani. Expander flows, geometric embeddings, and graph partitionings. In *Proc. of STOC*, pages 222–231, 2004.

[12] Y. Aumann and Y. Rabani. An $O(\log k)$ approximate min-cut max-flow theorem and approximation algorithm. *SIAM J. Comput.*, 27(1):291–301 (electronic), 1998.

[13] B. Babcock, S. Babu, M. Datar, R. Motwani, and J. Widom. Models and issues in data stream systems. In *Proc. of PODS*, pages 1–16, 2002.

[14] 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. of STOC*, pages 316–324, 2003.

[15] M. Charikar. Similarity estimation techniques from rounding. In *Proc. of STOC*, pages 380–388, 2002.

[16] M. Charikar and R. Krauthgamer. Embedding the ulam metric into ℓ_1. *Theory of Computing*, 2(11):207–224, 2006.

[17] E. Chávez, G. Navarro, R. Baeza-Yates, and J. Marroquin. Searching in metric spaces. *ACM Computing Surveys*, 33(3):273–321, Sept. 2001.

[18] G. Cormode and S. Muthukrishnan. An improved data stream summary: the count-min sketch and its applications. *J. Algorithms*, 55(1):58–75, 2005.

[19] G. Cormode and S. Muthukrishnan. Space efficient mining of multigraph streams. In *Proc. of PODS*, 2005.

[20] N. R. Devanur, S. A. Khot, R. Saket, and N. K. Vishnoi. Integrality gaps for sparsest cut and minimum linear arrangement problems. In *Proc. of STOC*, pages 537–546, 2006.

[21] F. Ergun, S. Kannan, R. Kumar, R. Rubinfeld, and M. Viswanathan. Spot-checkers. *J. Comput. Syst. Sci.*, 60(3):717–751, 2000.

[22] P. Flajolet and G. Martin. Probabilistic counting algorithms for data base applications. *J. Comput. Syst. Sci.*, 31:182–209, 1985.

[23] P. Gopalan, T. S. Jayram, R. Krauthgamer, and R. Kumar. Estimating the sortedness of a data stream. In *Proc. of SODA*, 2007.

[24] S. Har-Peled. A replacement for voronoi diagrams of near linear size. In *Proc. of FOCS*, pages 94–103, 2001.

[25] P. Indyk. On approximate nearest neighbors in l_∞ norm. *J. Comput. Syst. Sci., to appear.* Preliminary version appeared in FOCS'98.

[26] P. Indyk. Tutorial: Algorithmic applications of low-distortion geometric embeddings. *Proc. of FOCS*, pages 10–33, 2001.

[27] P. Indyk. Approximate nearest neighbor algorithms for frechet metric via product metrics. *Proc. of SoCG*, pages 102–106, 2002.

[28] P. Indyk. Approximate nearest neighbor under edit distance via product metrics. In *Proc. of SODA*, pages 646–650, 2004.

[29] P. Indyk. A near linear time constant factor approximation for euclidean bichromatic matching (cost). In *Proc. of SODA*, 2007.

[30] P. Indyk and J. Matoušek. Discrete metric spaces. *CRC Handbook of Discrete and Computational Geometry, 2nd edition*, 2003.

[31] P. Indyk and R. Motwani. Approximate nearest neighbor: towards removing the curse of dimensionality. *Proc. of STOC*, pages 604–613, 1998.

[32] P. Indyk and N. Thaper. Fast color image retrieval via embeddings. *Workshop on Statistical and Computational Theories of Vision (at ICCV)*, 2003.

[33] P. Indyk and D. Woodruff. Optimal approximations of the frequency moments of data streams. *Proc. of STOC*, 2005.

[34] T. Jayram and D. Woodruff. Cascaded stream aggregates. *Manuscript*, 2008.

[35] W. B. Johnson and J. Lindenstrauss, editors. *Handbook of the geometry of Banach spaces. Vol. I.* North-Holland Publishing Co., Amsterdam, 2001.

[36] S. Khot and A. Naor. Nonembeddability theorems via fourier analysis. *Math. Ann.*, 334(4):821–852, 2006. Preliminary version appeared in FOCS'05.

[37] S. A. Khot and N. K. Vishnoi. The unique games conjecture, integrality gap for cut problems and embeddability of negative type metrics into ℓ_1. *Proc. of FOCS*, 00:53–62, 2005.

[38] R. Krauthgamer and Y. Rabani. Improved lower bounds for embeddings into l_1. In *Proc. of SODA*, pages 1010–1017, 2006.

[39] N. Linial. Finite metric spaces - combinatorics, geometry and algorithms. *Proceedings of the International Congress of Mathematicians III*, pages 573–586, 2002.

[40] N. Linial, E. London, and Y. Rabinovich. The geometry of graphs and some of its algorithmic applications. *Combinatorica*, 15(2):215–245, 1995.

[41] J. I. Marden. *Analyzing and Modeling Rank Data*. Monographs on Statistics and Applied Probability 64. CRC Press, 1995.

[42] J. Matousek. *Lectures on Discrete Geometry*. Springer, 2002.

[43] A. Naor and G. Schechtman. Planar earthmover is not in l_1. *Proc. of FOCS*, 2006.

[44] R. Ostrovsky and Y. Rabani. Low distortion embedding for edit distance. *J. ACM*, 54(5), 2007. Preliminary version appeared in STOC'05.

[45] Y. Rubner, C. Tomassi, and L. J. Guibas. The earth mover's distance as a metric for image retrieval. *International Journal of Computer Vision*, 40(2):99–121, 2000.