Approximate Nearest Neighbor for Curves – Simple, Efficient, and Deterministic

Arnold Filtser
Department of Computer Science, Columbia University, New York, NY, USA
http://www.cs.columbia.edu/~arnoldf/
arnold273@gmail.com

Omrit Filtser
Department of Applied Mathematics and Statistics, Stony Brook University, NY, USA
https://omrit.filtser.com/
omrit.filtser@gmail.com

Matthew J. Katz
Department of Computer Science, Ben-Gurion University of the Negev, Beer Sheva, Israel
matya@cs.bgu.ac.il

Abstract

In the $(1+\varepsilon, r)$-*approximate near-neighbor* problem for curves (ANNC) under some similarity measure δ, the goal is to construct a data structure for a given set $\mathcal{C}$ of curves that supports approximate near-neighbor queries: Given a query curve Q, if there exists a curve $C \in \mathcal{C}$ such that $\delta(Q, C) \leq r$, then return a curve $C' \in \mathcal{C}$ with $\delta(Q, C') \leq (1+\varepsilon)r$. There exists an efficient reduction from the $(1+\varepsilon)$-*approximate nearest-neighbor* problem to ANNC, where in the former problem the answer to a query is a curve $C \in \mathcal{C}$ with $\delta(Q, C) \leq (1+\varepsilon) \cdot \delta(Q, C^*)$, where C^* is the curve of $\mathcal{C}$ most similar to Q.

Given a set $\mathcal{C}$ of n curves, each consisting of m points in d dimensions, we construct a data structure for ANNC that uses $n \cdot O(\frac{1}{\varepsilon})^{md}$ storage space and has $O(md)$ query time (for a query curve of length m), where the similarity measure between two curves is their discrete Fréchet or dynamic time warping distance. Our method is simple to implement, deterministic, and results in an exponential improvement in both query time and storage space compared to all previous bounds.

Further, we also consider the *asymmetric* version of ANNC, where the length of the query curves is $k \ll m$, and obtain essentially the same storage and query bounds as above, except that m is replaced by k. Finally, we apply our method to a version of approximate range counting for curves and achieve similar bounds.

2012 ACM Subject Classification Theory of computation → Computational geometry; Theory of computation → Design and analysis of algorithms

Keywords and phrases polygonal curves, Fréchet distance, dynamic time warping, approximation algorithms, (asymmetric) approximate nearest neighbor, range counting

Digital Object Identifier 10.4230/LIPIcs.ICALP.2020.48

Category Track A: Algorithms, Complexity and Games

Related Version https://arxiv.org/abs/1902.07562

Funding *Arnold Filtser*: Supported by the Simons Foundation.
Omrit Filtser: Supported by the Eric and Wendy Schmidt Fund for Strategic Innovation, by the Council for Higher Education of Israel, and by Ben-Gurion University of the Negev.
Matthew J. Katz: Supported by grant 1884/16 from the Israel Science Foundation.

Acknowledgements We wish to thank Boris Aronov for helpful discussions on the problems studied in this paper.

1 Introduction

Nearest neighbor search is a fundamental and well-studied problem that has various applications in machine learning, data analysis, and classification. This important task also arises in applications where the recorded instances are trajectories or polygonal curves modeling, for example, epigenetic and surgical processes, market value fluctuations, population growth, the number of the requests per hour received at some web-page, and even the response of a football player in a given situation.

Let $\mathcal{C}$ be a set of n curves, each consisting of at most m points in d dimensions, and let δ be some distance measure for curves. In the *nearest-neighbor* problem for curves, the goal is to construct a data structure for $\mathcal{C}$ that supports nearest-neighbor queries, that is, given a query curve Q of length at most m, return the curve $C^* \in \mathcal{C}$ closest to Q (according to δ). The approximation version of this problem is the $(1+\varepsilon)$-*approximate nearest-neighbor* problem, where the answer to a query Q is a curve $C \in \mathcal{C}$ with $\delta(Q,C) \leq (1+\varepsilon)\delta(Q,C^*)$. We study a decision version of this approximation problem, which is called the $(1+\varepsilon,r)$-*approximate near-neighbor* problem for curves (ANNC). Here, if there exists a curve in $\mathcal{C}$ that lies within distance r of the query curve Q, one has to return a curve in $\mathcal{C}$ that lies within distance $(1+\varepsilon)r$ of Q. Note that there exists a reduction from the $(1+\varepsilon)$-approximate nearest-neighbor problem to the $(1+\varepsilon,r)$-approximate near-neighbor problem [14, 22, 13], at the cost of an additional logarithmic factor in the query time and an $O(\log^2 n)$ factor in the storage space.

In practice, it is often the case that the query curves are significantly shorter than the input curves (e.g., Google-search queries). Thus, we also study the *asymmetric setting* of $(1+\varepsilon,r)$-ANNC, where each of the input curves has complexity at most m, while each query curve has complexity at most $k \ll m$.

There are many methods that are used in real-world applications for comparing curves, and one of the most prevalent is the (discrete) *Fréchet* distance (DFD for short), which is often described by the following analogy. Two frogs are hopping from vertex to vertex along two polygonal curves. At each step, one of the frogs or both frogs may advance to the next vertex on its curve. The discrete Fréchet distance is defined as the smallest maximum distance between the frogs that can be achieved in such a joint sequence of hops. Another useful distance measure for curves or time series is the *dynamic time warping* distance (DTW for short), in which instead of taking the smallest maximum distance we take the smallest sum of distances.

In the last several years, a series of papers have been written investigating the approximate near-neighbor problem for curves (ANNC) and its variants under the Fréchet distance [15, 6, 10, 7, 1, 12, 2] (see Table 1), and several different approaches and sophisticated methods were utilized in order to provide efficient data structures. Up to now, all data structures for ANNC under DFD have either an exponential in m query time, or an infeasible storage space bound. In this paper, for the first time, we manage to remove the exponential factor from the query time, while also significantly reducing the space consumption. Our approach consists of a discretization of space based on the input curves, which allows us to prepare a small set of curves that captures all possible queries approximately.

Indyk [15] was the first to give a deterministic near-neighbor data structure for curves under DFD. The data structure achieves an approximation factor of $O((\log m + \log\log n)^{t-1})$ given some trade-off parameter $t > 1$. Its space consumption is very high, $O(m^2|X|)^{tm^{1/t}} \cdot n^{2t}$, where $|X|$ is the size of the domain on which the curves are defined, and the query time is $(m \log n)^{O(t)}$. In Table 1 we set $t = 1 + o(1)$ to obtain a constant approximation factor.

Later, Driemel and Silvestri [10] presented a locality-sensitive-hashing scheme for curves under DFD, improving the result of Indyk for short curves. Their data structure uses $O(2^{4md}n\log n)$ space and answers queries in $O(2^{4md}\log n)$ time with an approximation factor of $O(d^{3/2})$. They also provide a trade-off between approximation quality and computational performance: for $d = O(1)$, and given a parameter $k \in [m]$, a data structure of size $O(2^{2k}m^{k-1}n\log n + mn)$ is constructed that answers queries in $O(2^{2k}m^k\log n)$ time with an approximation factor of $O(m/k)$. They also show that this result can be applied to DTW, but only for the extreme of the trade-off which gives an $O(m)$ approximation.

Recently, Emiris and Psarros [12] presented near-neighbor data structures for curves under both DFD and DTW. Their algorithm provides an approximation factor of $(1+\varepsilon)$, at the expense of increased space usage and preprocessing time. They use the idea that for a fixed alignment between two curves (i.e., a given sequence of hops of the two frogs), the problem can be reduced to the near-neighbor problem for points in ℓ_∞ product of ℓ_2 spaces. Their basic idea is to construct a data structure for every possible alignment. Once a query is given, they query all these data structures and return the closest curve found. This approach is responsible for the 2^m factor in their query time. Furthermore, they generalize this approach using randomized projections of ℓ_p-products of Euclidean metrics (for any $p \geq 1$), and define the $\ell_{p,2}$-distance for curves (for $p \geq 1$), which is exactly DFD when $p = \infty$, and DTW when $p = 1$ (see Section 2). The space used by their data structure is $\tilde{O}(n) \cdot (2 + \frac{d}{\log m})^{O(m^{1+1/\varepsilon} \cdot d\log(1/\varepsilon))}$ with query $\tilde{O}(dm^{1+1/\varepsilon} \cdot 2^{4m}\log n)$ for DFD and $\tilde{O}(n) \cdot \frac{1}{\varepsilon}^{O(md)}$ space and $\tilde{O}(d \cdot 2^{4m}\log n)$ query for DTW.

Subsequent work. In a recent manuscript, Driemel, Psarros, and Schmidt [9], study the asymmetric setting of $(1+\varepsilon, r)$-ANNC under DFD. They follow our approach of preparing in advance the answers to all relevant queries on a discretization of the space, to construct a randomized data structure with space in $n \cdot O\left(\frac{kd^{3/2}}{\varepsilon}\right)^{dk}$ and query time in $O(dk)$. They also show how to derandomize their data structure, at the cost of increasing the space to $d^{3/2}nk\varepsilon^{-1} \cdot O\left(\frac{kd^{3/2}}{\varepsilon}\right)^{dk}$, and the query time to $O(d^{5/2}k^2\varepsilon^{-1}(\log n + kd\log\frac{kd}{\varepsilon}))$. This provides additional evidence that our approach to ANNC, although quite simple and easy to implement, seems to produce more efficient data structures than those obtained using tools such as LSH and randomized projections. Moreover, in this version of our manuscript we show how to improve upon the results in [9] for the asymmetric setting.

Our results. We present a data structure for the $(1+\varepsilon, r)$-approximate near-neighbor problem using a bucketing method. We construct a relatively small set of curves $\mathcal{I}$ such that given a query curve Q, if there exists some curve in $\mathcal{C}$ within distance r of Q, then one of the curves in $\mathcal{I}$ must be very close to Q. The points of the curves in $\mathcal{I}$ are chosen from a simple discretization of space, thus, while it is not surprising that we get the best query time, it is surprising that we achieve a better space bound. Moreover, while the analysis of the space bounds is rather involved, the implementation of our data structures remain simple in practice.

See Table 1 for a summary of our results. In the table, we do not state our result for the general $\ell_{p,2}$-distance. Instead, we state our results for the two most important cases, i.e. DFD and DTW, and compare them with previous work. Note that our results substantially improve the current state of the art for any $p \geq 1$. In particular, we remove the exponential dependence on m in the query bounds and significantly improve the space bounds.

Our results for the asymmetric setting, where the query curve Q has complexity $k \ll m$, are summarized in Table 2. We show that in the asymmetric setting for DFD, our data structure can be slightly modified in order to achieve query time and storage space independent of m. Moreover, the storage space and query time matches those of the symmetric setting, by replacing m with k.

We also apply our methods to an approximation version of range counting for curves (for the general $\ell_{p,2}$ distance) and achieve bounds similar to those of our ANNC data structure. Moreover, at the cost of an additional $O(n)$-factor in the space bound, we can also answer the corresponding approximation version of range searching, thus answering a question of Afshani and Driemel [1], with respect to DFD.

We note that our approach with obvious modifications works also in a dynamic setting, that is, we can construct an efficient dynamic data structure for ANNC as well as for other related problems such as range counting and range reporting for curves.

Another significant advantage of our approach is that, unlike some of the previous solutions, our data structure always returns an answer, and never returns a curve at distance greater than $(1+\varepsilon)r$ from the query curve, i.e., there are no false positives. This is an important property of our solution, due to the fact that verifying the validity of the answer (i.e., computing the distance between two curves) cannot be done in strongly subquadratic time (assuming SETH, see [4]), which is already more than our query time (for $d < m$).

Table 1 Our approximate near-neighbor data structure under DFD and DTW compared to the previous results.

	Space	Query	Approx.	Comments
DFD	$O(m^2\lvert X\rvert)^{m^{1-o(1)}} \cdot n^{2+o(1)}$	$(m \log n)^{O(1)}$	$O(1)$	deterministic, [15]
	$O(2^{4md} n \log n)$	$O(2^{4md} \log n)$	$O(d^{3/2})$	randomized, using LSH [10]
	$\tilde{O}(n) \cdot (2 + \frac{d}{\log m})^{O(m^{1+1/\varepsilon} \cdot d \log(\frac{1}{\varepsilon}))}$	$\tilde{O}(dm^{1+1/\varepsilon} \cdot 2^{4m} \log n)$	$1+\varepsilon$	randomized, [12]
	$n \cdot O(\frac{1}{\varepsilon})^{md}$	$O(md)$	$1+\varepsilon$	deter. (rand. construction), Theorem 9
DTW	$O(n \log n + mn)$	$O(m \log n)$	$O(m)$	randomized, using LSH, $d = O(1)$, [10]
	$\tilde{O}(n) \cdot \frac{1}{\varepsilon}^{O(md)}$	$\tilde{O}(d \cdot 2^{4m} \log n)$	$1+\varepsilon$	randomized, [12]
	$n \cdot O(\frac{1}{\varepsilon})^{m(d+1)}$	$O(md)$	$1+\varepsilon$	deter. (rand. construction), Theorem 15

More related work. De Berg, Gudmundsson, and Mehrabi [7] described a dynamic data structure for approximate nearest neighbor for curves (which can also be used for other types of queries such as range reporting), under the (continuous) Fréchet distance. Their data structure uses $n \cdot O\left(\frac{1}{\varepsilon}\right)^{2m}$ space and has $O(m)$ query time, but with an additive error of $\varepsilon \cdot reach(Q)$, where $reach(Q)$ is the maximum distance between the start vertex of the query curve Q and any other vertex of Q. Furthermore, their query procedure might fail when the distance to the nearest neighbor is relatively large.

Afshani and Driemel [1] studied (exact) range searching under both the discrete and continuous Fréchet distance. In this problem, the goal is to preprocess $\mathcal{C}$ such that given a query curve Q of length m_q and a radius r, all the curves in $\mathcal{C}$ that are within distance r from

Table 2 Summary of previous and current results for the asymmetric approximate near-neighbor data structure for curves. All the results in the table are w.r.t. DFD. The approximation ratio is $1+\varepsilon$ for $\varepsilon \in (0,1)$, and our data structures always succeed. Historic note: [9] is a subsequent work to the first version of this paper arXiv:1902.07562. In this second version we also apply our counting techniques to the asymmetric cases.

Space	Query	Deterministic construction?	Reference
$n \cdot \left(O(\frac{kd^{3/2}}{\varepsilon})^{kd}\right)$	$O(kd)$	no	[9]
$n \cdot \left(O(\frac{kd^{3/2}}{\varepsilon})^{kd+1}\right)$	$O(\frac{k^2 d^{5/2}}{\varepsilon}(\log n + kd\log(\frac{kd}{\varepsilon})))$	yes	[9]
$n \cdot O(\frac{1}{\varepsilon})^{kd}$	$O(kd)$	no	Theorem 11
$n \cdot O(\frac{1}{\varepsilon})^{kd}$	$O(kd\log(\frac{nkd}{\varepsilon}))$	yes	Theorem 18

Q can be found efficiently. For DFD, their data structure uses $O(n(\log\log n)^{m-1})$ space and has $O(n^{1-\frac{1}{d}} \cdot \log^{O(m)} n \cdot m_q^{O(d)})$ query time, where m_q is limited to $\log^{O(1)} n$. Additionally, they provide a lower bound in the pointer model, stating that every data structure with $Q(n)+O(k)$ query time, where k is the output size, has to use roughly $\Omega\left((n/Q(n))^2\right)$ space in the worst case (even for $m_q = 1$). Afshani and Driemel conclude their paper by asking whether more efficient data structures might be constructed if one allows approximation.

De Berg, Cook IV, and Gudmundsson [6], considered the following range counting problem under the continuous Fréchet distance. Given a polygonal curve C with m vertices, they show how to preprocess it into a data structure of size $O(k \cdot \text{polylog(m)})$, so that, given a query segment s, one can return a constant approximation of the number of subcurves of C that lie within distance r of s in $O(\frac{m}{\sqrt{k}} \cdot \text{polylog(m)})$ time, where k is a parameter between m and m^2.

Aronov et al. [2] managed to obtain practical bounds for two cases of the asymmetric $(1+\varepsilon, r)$-ANNC under DFD: (i) when Q is a line segment (i.e., $k=2$), or (ii) when $\mathcal{C}$ consists of line segments (i.e., $m=2$). The bounds on the size of the data structure and query time are nearly linear in the size of the input and query curve, respectively. Specifically, for the case where $k=2$, they achieve query time $O(\log^4(\frac{n}{\varepsilon}))$ and storage space $O(n\frac{1}{\varepsilon^4}\log^4(\frac{n}{\varepsilon}))$. They also provide efficient data structures for several other variants of the problem: the (exact) NNC where ℓ_∞ is used for interpoint distances, and the case where the location of the input curves is only fixed up to translation.

1.1 Technical ideas

We use a discretization of the space, by laying a d-dimensional uniform grid with edge length $\frac{\varepsilon r}{\sqrt{d}}$. The main ingredient in our data structure is then a relatively small set $\mathcal{I}$ of curves defined by grid points, which represents all possible queries. For each curve in $\mathcal{I}$ we store an index of a close enough curve from the input set $\mathcal{C}$. Given a query Q sufficiently close to some curve in $\mathcal{C}$, we find a representative Q' in $\mathcal{I}$ by simply rounding Q's vertices and return the index of the curve stored for Q'.

Given a point $x \in \mathbb{R}^d$, the number of grid points that are within distance $(1+\varepsilon)r$ from x is bounded by $O(\frac{1}{\varepsilon})^d$ (Corollary 7). Thus, given a curve C of length m, the total number of grid points that are within distance $(1+\varepsilon)r$ from one of its vertices is $m \cdot O(\frac{1}{\varepsilon})^d$. Naively, the number of curves needed to represent all possible queries of length m within distance

r of C is bounded by the number of ways to choose m points with repetitions from a set of grid points of size $m \cdot O(\frac{1}{\varepsilon})^d$, which is bounded by $m^m \cdot O(\frac{1}{\varepsilon})^{md}$. This infeasible bound on the storage space might be the reason why more sophisticated solutions for ANNC have been suggested throughout the years.

One of the main technical contributions of this paper is an analysis leading to a significantly better bound, if we store only candidate curves that are within distance $(1+\varepsilon)r$ from C. Actually, in Section 3 we show that for the case of DFD, it is sufficient to store a set of representative curves of size only $O(\frac{1}{\varepsilon})^{md}$ for each input curve. The basic idea is to bound the number of representatives that can be obtained by some fixed alignment between C and the candidate curve (see Claim 8).

For the general case of $\ell_{p,2}$-distance (including DTW), we are minimizing the sum of distances instead of the maximum distance (as in DFD). Thus, we have to use a more dense grid (with edge length $\frac{\varepsilon r}{(2m)^{1/p}\sqrt{d}}$), and the situation becomes more complicated. First, unlike DFD, the triangle inequality does not hold for $\ell_{p,2}$-distance in general (including DTW). Second, since DFD is a min-max measure, the choice of different vertices for a representative curve is "independent" in a sense, whereas for $\ell_{p,2}$-distance in general, the choice of different vertices depends on their sum of distances from the input curve. Using more careful counting arguments and analysis of the alignment between two curves, we are able to show that in this case the number of representative curves that our data structure has to store per input curve is bounded by $O(\frac{1}{\varepsilon})^{m(d+1)}$ (see Claim 13).

To store the set $\mathcal{I}$ we simply use a dictionary, which can be implemented using a hash table and guarantees a query time linear in the size of the query. To obtain a fully deterministic solution, one can use a search tree instead. However, a naive implementation using a binary search tree results in an additional factor of $O(\log|\mathcal{I}|) = O(md\log(\frac{n}{\varepsilon}))$ to the query time, i.e., in a query time of $O(m^2d^2\log(\frac{n}{\varepsilon}))$. We show how to implement the dictionary using a prefix tree, exploiting the fact that the vertices of the curves in $\mathcal{I}$ are from a relatively small set of grid points, which improves the query time to $O(md\log(\frac{nmd}{\varepsilon}))$.

For the asymmetric setting (where the length of a query is $k \ll m$), we use simplifications of the input curves in order to obtain bounds that are independent of m. Given a curve C of length m, a simplification Π of C is a curve of length $k \ll m$ that is relatively close to C. Simplifications were used in order to provide approximate solutions in several asymmetric versions of problems on curves, such as clustering [5], and distance oracles [8, 9].

By the triangle inequality for DFD, every query curve Q within distance r from an input curve C is at distance at most $2r$ from the simplification Π (where Π is within distance r from C). Thus, it is enough to prepare for query curves at distance at most $2r$ from Π, which follows from previous arguments. Note that the query time and storage space are independent of m.

2 Preliminaries

To simplify the presentation, we assume throughout the paper that all the input curves have exactly the same size, m, and all the query curves have exactly the same size, either m or k, depending on whether we are considering the standard or the asymmetric version. This assumption can be easily removed (see Remark 16 at the end of Section 5).

Let $\mathcal{C}$ be a set of n curves, each consisting of m points in d dimensions, and let δ be some distance measure for curves.

▶ **Problem 1** ($(1+\varepsilon)$-approximate nearest-neighbor for curves). *Given a parameter $0 < \varepsilon \leq 1$, preprocess $\mathcal{C}$ into a data structure that given a query curve Q, returns a curve $C' \in \mathcal{C}$, such that $\delta(Q, C') \leq (1+\varepsilon) \cdot \delta(Q, C)$, where C is the curve in $\mathcal{C}$ closest to Q.*

▶ **Problem 2** ($(1+\varepsilon, r)$-approximate near-neighbor for curves). *Given a parameter r and $0 < \varepsilon \leq 1$, preprocess $\mathcal{C}$ into a data structure that given a query curve Q, if there exists a curve $C_i \in \mathcal{C}$ such that $\delta(Q, C_i) \leq r$, returns a curve $C_j \in \mathcal{C}$ such that $\delta(Q, C_j) \leq (1+\varepsilon)r$.*

▶ **Problem 3** (Asymmetric $(1+\varepsilon, r)$-approximate near-neighbor for curves). *Given parameters r,k, and $0 < \varepsilon \leq 1$, preprocess $\mathcal{C}$ into a data structure that given a query curve Q of length k, if there exists a curve $C_i \in \mathcal{C}$ such that $\delta(Q, C_i) \leq r$, returns a curve $C_j \in \mathcal{C}$ such that $\delta(Q, C_j) \leq (1+\varepsilon)r$.*

Curve alignment. Given two integers m_1, m_2, let $\tau := \langle (i_1, j_1), \ldots, (i_t, j_t) \rangle$ be a sequence of pairs where $i_1 = j_1 = 1$, $i_t = m_1$,$j_t = m_2$, and for each $1 < k \leq t$, one of the following conditions holds:

(i) $i_k = i_{k-1} + 1$ and $j_k = j_{k-1}$,
(ii) $i_k = i_{k-1}$ and $j_k = j_{k-1} + 1$, or
(iii) $i_k = i_{k-1} + 1$ and $j_k = j_{k-1} + 1$.

We call such a sequence τ an *alignment* of two curves.

Let $P = (p_1, \ldots, p_{m_1})$ and $Q = (q_1, \ldots, q_{m_2})$ be two curves of lengths m_1 and m_2, respectively, in d dimensions. We say that an alignment τ w.r.t. P and Q matches p_i and p_j if $(i, j) \in \tau$.

Discrete Fréchet distance (DFD). The *Fréchet cost* of an alignment τ w.r.t. P and Q is $\sigma_{dF}(\tau(P, Q)) := \max_{(i,j)\in\tau} \|p_i - q_j\|_2$. The discrete Fréchet distance is defined over the set $\mathcal{T}$ of all alignments as

$$d_{dF}(P, Q) = \min_{\tau\in\mathcal{T}} \sigma_{dF}(\tau(P, Q)).$$

Dynamic time wrapping (DTW). The *time warping cost* of an alignment τ w.r.t. P and Q is $\sigma_{DTW}(\tau(P, Q)) := \sum_{(i,j)\in\tau} \|p_i - q_j\|_2$. The DTW distance is defined over the set $\mathcal{T}$ of all alignments as

$$d_{DTW}(P, Q) = \min_{\tau\in\mathcal{T}} \sigma_{DTW}(\tau(P, Q)).$$

$\ell_{p,2}$-distance for curves. The $\ell_{p,2}$-*cost* of an alignment τ w.r.t. P and Q is $\sigma_{p,2}(\tau(P, Q)) := \left(\sum_{(i,j)\in\tau} \|p_i - q_j\|_2^p\right)^{1/p}$. The $\ell_{p,2}$-distance between P and Q is defined over the set $\mathcal{T}$ of all alignments as

$$d_{p,2}(P, Q) = \min_{\tau\in\mathcal{T}} \sigma_{p,2}(\tau(P, Q)).$$

Notice that $\ell_{p,2}$-distance is a generalization of DFD and DTW, in the sense that $\sigma_{dF} = \sigma_{\infty,2}$ and $d_{dF} = d_{\infty,2}$, $\sigma_{DTW} = \sigma_{1,2}$ and $d_{DTW} = d_{1,2}$. Also note that DFD satisfies the triangle inequality, but DTW and $\ell_{p,2}$-distance (for $p \neq \infty$) do not (see Section 5 for details).

Emiris and Psarros [12] showed that the number of all possible alignments of two curves is in $O(m \cdot 2^{2m})$. We reduce this bound by counting only alignments that can determine the $\ell_{p,2}$-distance between two curves.[1] More formally, let τ be an alignment. If there

[1] Since our storage space is already in $O(\frac{1}{\varepsilon})^{md}$, and $m \cdot 2^{2m} \leq 3^{2m}$ is in $O(1)^{md}$, we could have used this larger upper bound. However, in Lemma 4 we show a tight upper bound on the number of relevant alignments, which may be useful for other applications.

exists an alignment τ' such that $\tau' \subset \tau$, then clearly $\sigma_{p,2}(\tau'(P,Q)) \leq \sigma_{p,2}(\tau(P,Q))$, for any $1 \leq p \leq \infty$ and for any two curves P and Q. In this case, we say that τ cannot determine the $\ell_{p,2}$-distance between two curves.

▶ **Lemma 4.** *The number of different alignments that can determine the $\ell_{p,2}$-distance between two m-curves (for any $1 \leq p \leq \infty$) is at most $O(\frac{2^{2m}}{\sqrt{m}})$.*

Proof. Let $\tau = \langle (i_1, j_1), \ldots, (i_t, j_t) \rangle$ be an alignment. Notice that $m \leq t \leq 2m-1$. By definition, τ has 3 types of (consecutive) subsequences of length two:

(i) $\langle (i_k, j_k), (i_k+1, j_k) \rangle$,
(ii) $\langle (i_k, j_k), (i_k, j_k+1) \rangle$, and
(iii) $\langle (i_k, j_k), (i_k+1, j_k+1) \rangle$.

Denote by $\mathcal{T}_1$ the set of all alignments that do not contain any subsequence of type (iii). Then, any $\tau_1 \in \mathcal{T}_1$ is of length exactly $2m-1$. Moreover, τ_1 contains exactly $2m-2$ subsequences of length two, of which $m-1$ are of type (i) and $m-1$ are of type (ii). Therefore, $|\mathcal{T}_1| = \binom{2m-2}{m-1} = O(\frac{2^{2m}}{\sqrt{m}})$.

Assume that an alignment τ contains a subsequence of the form $(i_k, j_k-1), (i_k, j_k), (i_k+1, j_k)$, for some $1 < k \leq t-1$. Notice that removing the pair (i_k, j_k) from τ results in a legal alignment τ', such that $\sigma_{p,2}(\tau'(P,Q)) \leq \sigma_{p,2}(\tau(P,Q))$, for any $1 \leq p \leq \infty$ and two curves P, Q. We call the pair (i_k, j_k) a *redundant pair*. Similarly, if τ contains a subsequence of the form $(i_k-1, j_k), (i_k, j_k), (i_k, j_k+1)$, for some $1 < k \leq t-1$, then the pair (i_k, j_k) is also a redundant pair. Therefore we only care about alignments that do not contain any redundant pairs. Denote by $\mathcal{T}_2$ the set of all alignments that do not contain any redundant pairs, then any $\tau_2 \in \mathcal{T}_2$ contains at least one subsequence of type (iii).

We claim that for any alignment $\tau_2 \in \mathcal{T}_2$, there exists a unique alignment $\tau_1 \in \mathcal{T}_1$. Indeed, if we add the redundant pair (i_l, j_l+1) between (i_l, j_l) and (i_l+1, j_l+1) for each subsequence of type (iii) in τ_2, we obtain an alignment $\tau_1 \in \mathcal{T}_1$. Moreover, since τ_2 does not contain any redundant pairs, the reverse operation on τ_1 results in τ_2. Thus we obtain $|\mathcal{T}_2| \leq |\mathcal{T}_1| = O(\frac{2^{2m}}{\sqrt{m}})$. ◀

Points and balls. Given a point $x \in \mathbb{R}^d$ and a real number $R > 0$, we denote by $B_p^d(x, R)$ the d-dimensional ball under the ℓ_p norm with center x and radius R, i.e., a point $y \in \mathbb{R}^d$ is in $B_p^d(x, R)$ if and only if $\|x - y\|_p \leq R$, where $\|x - y\|_p = \left(\sum_{i=1}^d |x_i - y_i|^p\right)^{1/p}$. Let $B_p^d(R) = B_p^d(\mathbf{0}, R)$, and let $V_p^d(R)$ be the volume (w.r.t. Lebesgue measure) of $B_p^d(R)$, then

$$V_p^d(R) = \frac{2^d \Gamma(1+1/p)^d}{\Gamma(1+d/p)} R^d,$$

where $\Gamma(\cdot)$ is Euler's Gamma function (an extension of the factorial function). For $p = 2$ and $p = 1$, we get

$$V_2^d(R) = \frac{\pi^{d/2}}{\Gamma(1+d/2)} R^d \quad \text{and} \quad V_1^d(R) = \frac{2^d}{d!} R^d.$$

Our approach consists of a discretization of the space using lattice points, i.e., points from $\mathbb{Z}^d$.

▶ **Lemma 5.** *The number of lattice points in the d-dimensional ball of radius R under the ℓ_p norm (i.e., in $B_p^d(R)$) is bounded by $V_p^d(R + d^{1/p})$.*

Proof. With each lattice point $z = (z_1, z_2, \ldots, z_d)$, $z_i \in \mathbb{Z}$, we match the d-dimensional lattice cube $C(z) = [z_1, z_1+1] \times [z_2, z_2+1] \times \cdots \times [z_d, z_d+1]$. Notice that $z \in C(z)$, and the ℓ_p-diameter of a lattice cube is $d^{1/p}$. Therefore, the number of lattice points in the ℓ_p^d-ball of radius R is bounded by the number of lattice cubes that are contained in a ℓ_p^d-ball with radius $R + d^{1/p}$. This number is bounded by $V_p^d(R + d^{1/p})$ divided by the volume of a lattice cube, which is $1^d = 1$. ◀

▶ Remark 6. In general, in all our data structures we do not assume any bound on the dimension d. However, using dimension reduction techniques, we may assume that $d \leq O(\frac{\log(nm)}{\varepsilon^2})$. See Section 9 for details.

3 Discrete Fréchet distance (DFD)

Consider the infinite d-dimensional grid with edge length $\frac{\varepsilon r}{\sqrt{d}}$. Given a point x in $\mathbb{R}^d$, by rounding one can find in $O(d)$ time the grid point x' closest to x, and $\|x - x'\|_2 \leq \frac{\varepsilon r}{2}$. Let $G(x, R)$ denote the set of grid points that are contained in $B_2^d(x, R)$.

▶ **Corollary 7.** $|G(x, (1+\varepsilon)r)| = O(\frac{1}{\varepsilon})^d$.

Proof. We scale our grid so that the edge length is 1, hence we are looking for the number of lattice points in $B_2^d(x, \frac{1+\varepsilon}{\varepsilon}\sqrt{d})$. By Lemma 5 we get that this number is bounded by the volume of the d-dimensional ball of radius $\frac{1+\varepsilon}{\varepsilon}\sqrt{d} + \sqrt{d} \leq \frac{3\sqrt{d}}{\varepsilon}$. Using Stirling's formula we conclude that

$$V_2^d\left(\frac{3\sqrt{d}}{\varepsilon}\right) = \frac{\pi^{\frac{d}{2}}}{\Gamma(\frac{d}{2}+1)} \cdot \left(\frac{3\sqrt{d}}{\varepsilon}\right)^d \leq \left(\frac{\alpha}{\varepsilon}\right)^d,$$

where α is a constant. For example, if d is even, then

$$V_2^d\left(\frac{3\sqrt{d}}{\varepsilon}\right) = \frac{\pi^{\frac{d}{2}}}{(\frac{d}{2})!} \cdot \left(\frac{3\sqrt{d}}{\varepsilon}\right)^d \leq \frac{\pi^{\frac{d}{2}}}{\sqrt{2\pi}(d/2)^{d/2+1/2}e^{-d/2}} \cdot \left(\frac{3\sqrt{d}}{\varepsilon}\right)^d \leq \left(\frac{12.4}{\varepsilon}\right)^d = O\left(\frac{1}{\varepsilon}\right)^d.$$

◀

Denote by p_j^i the j'th point of C_i, and let $G_i = \bigcup_{1 \leq j \leq m} G(p_j^i, (1+\varepsilon)r)$ and $\mathcal{G} = \bigcup_{1 \leq i \leq n} G_i$, then by the above corollary we have $|G_i| = m \cdot O(\frac{1}{\varepsilon})^d$ and $|\mathcal{G}| = mn \cdot O(\frac{1}{\varepsilon})^d$. Let $\mathcal{I}_i$ be the set of all curves $\overline{Q} = (x_1, x_2, \ldots, x_m)$ with points from G_i, such that $d_{dF}(C_i, \overline{Q}) \leq (1 + \frac{\varepsilon}{2})r$.

▷ Claim 8. $|\mathcal{I}_i| = O(\frac{1}{\varepsilon})^{md}$ and it can be computed in $O(\frac{1}{\varepsilon})^{md}$ time.

Proof. Let $\overline{Q} \in \mathcal{I}_i$ and let τ be an alignment with $\sigma_{dF}(\tau(C_i, \overline{Q})) \leq (1 + \frac{\varepsilon}{2})r$. For each $1 \leq k \leq m$ let j_k be the smallest index such that $(j_k, k) \in \tau$. In other words, j_k is the smallest index that is matched to k by the alignment τ. Since $d_{dF}(C_i, \overline{Q}) \leq (1 + \frac{\varepsilon}{2})r$, we have $x_k \in B_2^d(p_{j_k}^i, (1 + \frac{\varepsilon}{2})r)$, for $k = 1, \ldots, m$. This means that for any curve $\overline{Q} \in \mathcal{I}_i$ such that $\sigma_{dF}(\tau(C_i, \overline{Q})) \leq (1 + \frac{\varepsilon}{2})r$, we have $x_k \in G(p_{j_k}^i, (1 + \frac{\varepsilon}{2})r)$, for $k = 1, \ldots, m$. By Corollary 7, the number of ways to choose a grid point x_k from $G(p_{j_k}^i, (1 + \frac{\varepsilon}{2})r)$ is bounded by $O(\frac{1}{\varepsilon})^d$.

We conclude that given an alignment τ, the number of curves $\overline{Q}$ with m points from G_i such that $\sigma_{dF}(\tau(C_i, \overline{Q})) \leq (1 + \frac{\varepsilon}{2})r$ is bounded by $O(\frac{1}{\varepsilon})^{md}$. Finally, by Lemma 4, the total number of curves in $\mathcal{I}_i$ is bounded by $2^{2m} \cdot O(\frac{1}{\varepsilon})^{md} = O(\frac{1}{\varepsilon})^{md}$.

To construct $\mathcal{I}_i$ we compute, for each of the $O(\frac{1}{\varepsilon})^{md}$ candidates, its discrete Fréchet distance to C_i. Thus, we construct $\mathcal{I}_i$ in total time $O(\frac{1}{\varepsilon})^{md} \cdot O(m^2) = O(\frac{1}{\varepsilon})^{md}$. (The latter equality is true, since clearly $(\frac{\alpha}{\varepsilon})^{md} \cdot O(m^2) \leq (\frac{c\alpha}{\varepsilon})^{md}$, i.e., $O(m^2) \leq c^{md}$, where α is the constant from Corollary 7 and $c > 1$ is a sufficiently large constant.) ◁

The data structure. Denote $\mathcal{I} = \bigcup_{1 \le i \le n} \mathcal{I}_i$, so $|\mathcal{I}| \le n \cdot O(\frac{1}{\varepsilon})^{md}$ and we construct $\mathcal{I}$ in total time $n \cdot O(\frac{1}{\varepsilon})^{md}$. Next, we would like to store the set $\mathcal{I}$ in a dictionary (a hash table or a lookup table) $\mathcal{D}$, such that given a query curve Q, one can find Q in $\mathcal{D}$ (if it exists) in $O(md)$ time. We use Cuckoo Hashing [21] to construct a (dynamic) dictionary of linear space, constant worst-case query and deletion time, and constant expected amortized insertion time. We insert the curves of $\mathcal{I}$ into the dictionary $\mathcal{D}$ as follows. For each $1 \le i \le n$ and curve $\overline{Q} \in \mathcal{I}_i$, if $\overline{Q} \notin \mathcal{D}$, insert $\overline{Q}$ into $\mathcal{D}$, and set $C(\overline{Q}) \leftarrow C_i$. The storage space required for $\mathcal{D}$ is $O(|\mathcal{I}|)$, and to construct it we perform $|\mathcal{I}|$ insertions and look-up operations which take in total $O(|\mathcal{I}| \cdot md) = O(|\mathcal{I}|)$ expected time.

The query algorithm. Let $Q = (q_1, \ldots, q_m)$ be the query curve. The query algorithm is as follows: For each $1 \le k \le m$ find the grid point q'_k (not necessarily from $\mathcal{G}$) closest to q_k. This can be done in $O(md)$ time by rounding. Then, search for the curve $Q' = (q'_1, \ldots, q'_m)$ in the dictionary $\mathcal{D}$. If Q' is in $\mathcal{D}$, return $C(Q')$, otherwise, return NO. The total query time is then $O(md)$.

Correctness. Consider a query curve $Q = (q_1, \ldots, q_m)$. Assume that there exists a curve $C_i \in \mathcal{C}$ such that $d_{dF}(C_i, Q) \le r$. We show that the query algorithm returns a curve C^* with $d_{dF}(C^*, Q) \le (1+\varepsilon)r$.

Consider a point $q_k \in Q$. Denote by $q'_k \in \mathcal{G}$ the grid point closest to q_k, and let $Q' = (q'_1, \ldots, q'_m)$. We have $\|q_k - q'_k\|_2 \le \frac{\varepsilon r}{2}$, so $d_{dF}(Q, Q') \le \frac{\varepsilon r}{2}$. By the triangle inequality,

$$d_{dF}(C_i, Q') \le d_{dF}(C_i, Q) + d_{dF}(Q, Q') \le r + \frac{\varepsilon r}{2} = (1 + \frac{\varepsilon}{2})r,$$

so Q' is in $\mathcal{I}_i \subseteq \mathcal{I}$. This means that $\mathcal{D}$ contains Q' with a curve $C(Q') \in \mathcal{C}$ such that $d_{dF}(C(Q'), Q') \le (1 + \frac{\varepsilon}{2})r$, and the query algorithm returns $C(Q')$. Now, again by the triangle inequality,

$$d_{dF}(C(Q'), Q) \le d_{dF}(C(Q'), Q') + d_{dF}(Q', Q) \le (1 + \frac{\varepsilon}{2})r + \frac{\varepsilon r}{2} = (1+\varepsilon)r.$$

We obtain the following theorem.

▶ **Theorem 9.** *There exists a data structure for the $(1+\varepsilon, r)$-ANNC under DFD, with $n \cdot O(\frac{1}{\varepsilon})^{md}$ space, $n \cdot O(\frac{1}{\varepsilon})^{md}$ expected preprocessing time, and $O(md)$ query time.*

Table 3 Comparing our ANN data structure to previous structures, for a fixed ε (say $\varepsilon = 1/2$).

m	Reference	Space	Query	Approx.
$\log n$	[10]	$O(n^{4d+1} \log n)$	$\tilde{O}(n^{4d})$	$d\sqrt{d}$
	[12]	$n^{\Omega(d \log n)}$	$\tilde{O}(dn^4)$	$1+\varepsilon$
	Theorem 9	$n^{O(d)}$	$O(d \log n)$	$1+\varepsilon$
$O(1)$	[10]	$2^{O(d)} n \log n$	$2^{O(d)} \cdot \log n$	$d\sqrt{d}$
	[12]	$d^{O(d)}\tilde{O}(n)$	$O(d \log n)$	$1+\varepsilon$
	Theorem 9	$2^{O(d)} n$	$O(d)$	$1+\varepsilon$

4 The asymmetric setting under DFD

In this section, we show how to easily adapt our data structure to the asymmetric setting, by using simplifications of length at most k instead of the original input curves.

Bereg et al. [3] showed that given a curve C consisting of m points in 3D, and a parameter $r > 0$, there is an algorithm that runs in $O(m \log m)$ time and returns a simplification Π with minimum number of vertices such that $d_{dF}(C, \Pi) \leq r$. Their algorithm generalizes to higher dimensions, using an approximation algorithm for the minimum enclosing ball problem (see Kumar et al. [17]). In this section, we use the following generalization of their original approach ([3], Theorem 1). More details are given in Section 8.

▶ **Lemma 10.** *Let C be a curve consisting of m points in $\mathbb{R}^d$. Given parameters $k \leq m$, $r > 0$, and $\varepsilon \in (0, 1]$, there is an algorithm that runs in $O\left(\frac{d \cdot m \log m}{\varepsilon} + m \cdot \text{poly}\frac{1}{\varepsilon}\right)$ time that either returns a simplification Π consisting of k points such that $d_{dF}(C, \Pi) \leq (1+\varepsilon)r$, or declares that for every simplification Π with k points, it holds that $d_{dF}(C, \Pi) > r$.*

For each $C_i \in \mathcal{C}$, using Lemma 10 with parameter $\varepsilon = 1$, we find a curve Π_i of length k such that $d_{dF}(C_i, \Pi_i) \leq 2r$. If we fail to find such a curve, then we can ignore C_i, because it means that $d_{dF}(Q, C_i) > r$ for any curve Q of length k.

To reduce the space consumption of our data structure, we only store candidate curves of length k that are close enough to the simplifications Π_i. However, since the distance between the simplification Π_i and the input curve C_i could be up to $2r$, storing the answers for the set of candidate curves that are within distance $(1 + \frac{\varepsilon}{2})r$ from Π_i is not enough, because a query Q that is within distance $(1+\varepsilon)r$ from C_i might be as far as $(3+\varepsilon)r$ from Π_i. Thus, instead, we insert into our data structure all the curves that are within distance $4r$ from Π_i. This allows us to capture all query curves that are within distance r from C_i.

The data structure. We construct our data structure for the original (symmetric) version, with the following modifications. The set of input curves is $\mathcal{P} = \{\Pi_1, \ldots, \Pi_n\}$ (instead of $\mathcal{C}$), and the radius parameter is $4r$ (instead of r), but the grid edge length remains $\frac{\varepsilon r}{\sqrt{d}}$. In addition, we let $\mathcal{I}'_i$ be the set of all curves $\overline{Q}$ with k points from G_i, such that $d_{dF}(\overline{Q}, \Pi_i) \leq 4r$, and $\mathcal{I}_i$ will be the set of all curves $\overline{Q} \in \mathcal{I}'_i$ such that $d_{dF}(\overline{Q}, C_i) \leq (1 + \frac{\varepsilon}{2})r$. We insert the curves in $\mathcal{I}_i$ into the database $\mathcal{D}$ as before: For each $\overline{Q} \in \mathcal{I}_i$, if $\overline{Q} \notin \mathcal{D}$, insert $\overline{Q}$ into $\mathcal{D}$ and set $C(\overline{Q}) \leftarrow C_i$.

Notice that using $4r$ instead of r, increases the ratio between the radius and the grid edge length by only a factor of 4, and therefore the bound on $|\mathcal{I}'_i|$ does not change, except that m is replaced by k. Therefore, the bounds on the storage space and query time are similar to those of the original data structure, where m is replaced by k. Thus, the storage space is in $n \cdot O(\frac{1}{\varepsilon})^{kd}$ and the query time is in $O(kd)$. As for the preprocessing time, we get an additional term of $O(nmd \log m)$ for computing the simplifications $\Pi_1, \ldots, \Pi_n$. We also need to compute the distances $d_{dF}(C_i, \overline{Q})$ in the construction of $\mathcal{I}_i$, for $1 \leq i \leq n$, which takes $n \cdot O(\frac{1}{\varepsilon})^{kd} \cdot O(mkd) = nm \cdot O(\frac{1}{\varepsilon})^{kd}$ time in total (as $kd \leq 2^{kd}$). Thus the total expected preprocessing time is $O(nmd \log m) + nm \cdot O(\frac{1}{\varepsilon})^{kd} = nm \cdot \left(O(d \log m) + O(\frac{1}{\varepsilon})^{kd}\right)$.

Correctness. Consider a query curve Q, and assume that there exists a curve $C_i \in \mathcal{C}$ such that $d_{dF}(C_i, Q) \leq r$. Then, Π_i is a curve of length k and $d_{dF}(C_i, \Pi_i) \leq 2r$. As in the previous section, let Q' be the curve computed by the query algorithm, then $d_{dF}(Q', Q) \leq \frac{\varepsilon r}{2}$. By the triangle inequality, we have $d_{dF}(Q', C_i) \leq d_{dF}(Q', Q) + d_{dF}(Q, C_i) \leq (1 + \frac{\varepsilon}{2})r$, and

$$d_{dF}(Q', \Pi_i) \leq d_{dF}(Q', C_i) + d_{dF}(C_i, \Pi_i) \leq (1 + \frac{\varepsilon}{2})r + 2r \leq 4r.$$

Therefore our data structure contains Q', and the query algorithm returns $C(Q')$, where $d_{dF}(C(Q'), Q') \leq (1+\frac{\varepsilon}{2})r$. Finally, again by the triangle inequality, we have

$$d_{dF}(C(Q'), Q) \leq d_{dF}(C(Q'), Q') + d_{dF}(Q', Q) \leq (1+\frac{\varepsilon}{2})r + \frac{\varepsilon r}{2} = (1+\varepsilon)r.$$

We obtain the following theorem.

▶ **Theorem 11.** *There exists a data structure for the asymmetric $(1+\varepsilon, r)$-ANNC under DFD, with $n \cdot O(\frac{1}{\varepsilon})^{dk}$ space, $nm \cdot \left(O(d \log m) + O(\frac{1}{\varepsilon})^{kd}\right)$ expected preprocessing time, and $O(kd)$ query time.*

5 $\ell_{p,2}$-distance of polygonal curves

For the near-neighbor problem under the $\ell_{p,2}$-distance, we use the same basic approach as in Section 3, but with two small modifications. The first is that we set the grid's edge length to $\frac{\varepsilon r}{(2m)^{1/p}\sqrt{d}}$, and redefine $G(x,R)$, G_i, and $\mathcal{G}$, as in Section 3 but with respect to the new edge length of our grid. The second modification is that we redefine $\mathcal{I}_i$ to be the set of all curves $\overline{Q} = (x_1, x_2, \ldots, x_m)$ with points from $\mathcal{G}$, such that $d_{p,2}(C_i, \overline{Q}) \leq (1+\frac{\varepsilon}{2})r$.

We assume without loss of generality from now and to the end of this section that $r = 1$ (we can simply scale the entire space by $1/r$), so the grid's edge length is $\frac{\varepsilon}{(2m)^{1/p}\sqrt{d}}$. The following corollary is respective to Corollary 7.

▶ **Corollary 12.** $|G(x,R)| = O\left(1 + \frac{m^{1/p}}{\varepsilon}R\right)^d$.

Proof. We scale our grid so that the edge length is 1, hence we are looking for the number of lattice points in $B_2^d(x, \frac{(2m)^{1/p}\sqrt{d}}{\varepsilon}R)$. By Lemma 5 we get that this number is bounded by the volume of the d-dimensional ball of radius $(1+\frac{(2m)^{1/p}}{\varepsilon}R)\sqrt{d}$. Using Stirling's formula we conclude,

$$V_2^d\left(\left(1+\frac{(2m)^{1/p}}{\varepsilon}R\right)\sqrt{d}\right) = \frac{\pi^{\frac{d}{2}}}{\Gamma(\frac{d}{2}+1)} \cdot \left(\left(1+\frac{(2m)^{1/p}}{\varepsilon}R\right)\sqrt{d}\right)^d = \alpha^d \cdot \left(1+\frac{m^{1/p}}{\varepsilon}R\right)^d$$

where α is a constant (approximately $4.13 \cdot 2^{1/p}$). ◀

In the following claim we bound the size of $\mathcal{I}_i$, which, surprisingly, is independent of p.

▷ Claim 13. $|\mathcal{I}_i| = O(\frac{1}{\varepsilon})^{m(d+1)}$ and it can be computed in $O(\frac{1}{\varepsilon})^{m(d+1)}$ time.

Proof. Let $\overline{Q} = (x_1, x_2, \ldots, x_m) \in \mathcal{I}_i$, and let τ be an alignment with $\sigma_{p,2}(\tau(C_i, \overline{Q})) \leq (1+\frac{\varepsilon}{2})$. For each $1 \leq k \leq m$ let j_k be the smallest index such that $(j_k, k) \in \tau$. In other words, j_k is the smallest index that is matched to k by the alignment τ.

Set $R_k = \|x_k - p^i_{j_k}\|_2$, then we have $\|(R_1, \ldots, R_m)\|_p \leq \sigma_{p,2}(\tau(C_i, \overline{Q})) \leq (1+\frac{\varepsilon}{2})$.

Let $\alpha_k = \left\lceil \frac{m^{1/p}}{\varepsilon}R_k \right\rceil$. By triangle inequality,

$$\begin{aligned}\|(\alpha_1, \alpha_2, \ldots, \alpha_m)\|_p &\leq \frac{m^{1/p}}{\varepsilon}\|(R_1, R_2, \ldots, R_m)\|_p + m^{1/p} \\ &\leq \frac{m^{1/p}}{\varepsilon}\left(1+\frac{\varepsilon}{2}\right) + m^{1/p} < \left(2+\frac{1}{\varepsilon}\right)m^{1/p}.\end{aligned}$$

Clearly, $x_k \in B_2^d(p^i_{j_k}, \alpha_k \frac{\varepsilon}{m^{1/p}})$.

We conclude that for each curve $\overline{Q} = (x_1, x_2, \ldots, x_m) \in \mathcal{I}_i$ there exists an alignment τ such that $\sigma_{p,2}(\tau(C_i, \overline{Q})) \leq 1 + \frac{\varepsilon}{2}$, and a sequence of integers $(\alpha_1, \ldots, \alpha_m)$ such that $\|(\alpha_1, \alpha_2, \ldots, \alpha_m)\|_p \leq (2 + \frac{1}{\varepsilon})m^{1/p}$ and $x_k \in B_2^d(p^i_{j_k}, \alpha_k \frac{\varepsilon}{m^{1/p}})$, for $k = 1, \ldots, m$. Therefore, the number of curves in $\mathcal{I}_i$ is bounded by the multiplication of three numbers:

1. The number of alignments that can determine the distance, which is at most 2^{2m} by Lemma 4.
2. The number of ways to choose a sequence of m positive integers $\alpha_1, \ldots, \alpha_m$ such that $\|(\alpha_1, \alpha_2, \ldots, \alpha_m)\|_p \leq (2 + \frac{1}{\varepsilon})m^{1/p}$, which is bounded by the number of lattice points in $B_p^m((2 + \frac{1}{\varepsilon})m^{1/p})$ (the m-dimensional ℓ_p-ball of radius $(2 + \frac{1}{\varepsilon})m^{1/p}$). By Lemma 5, this number is bounded by

$$V_p^m((2 + \frac{1}{\varepsilon})m^{1/p} + m^{1/p}) \leq V_p^m(\frac{4m^{1/p}}{\varepsilon}) = \frac{2^m \Gamma(1 + 1/p)^m}{\Gamma(1 + m/p)} \left(\frac{4m^{1/p}}{\varepsilon}\right)^m = O(\frac{1}{\varepsilon})^m \ ,$$

   where the last equality follows as $\frac{m^{m/p}}{\Gamma(1+m/p)} = O(1)^m$.
3. The number of ways to choose a curve $(x_1, x_2, \ldots, x_m)$, such that $x_k \in G(p^i_{j_k}, \alpha_k \frac{\varepsilon}{m^{1/p}})$, for $k = 1, \ldots, m$. By Corollary 12, the number of grid points in $G(p^i_{j_k}, \alpha_k \frac{\varepsilon}{m^{1/p}})$ is $O(1+\alpha_k)^d$, so the number of ways to choose $(x_1, x_2, \ldots, x_m)$ is at most $\Pi_{k=1}^m O(1+\alpha_k)^d = O(1)^{md} \left(\Pi_{k=1}^m (1 + \alpha_k)\right)^d$. By the inequality of arithmetic and geometric means we have

$$\begin{aligned}
(\Pi_{k=1}^m (1 + \alpha_k)^p)^{1/p} &\leq \left(\frac{\sum_{k=1}^m (1 + \alpha_k)^p}{m}\right)^{m/p} = \left(\frac{\|(1 + \alpha_1, \ldots, 1 + \alpha_m)\|_p}{m^{1/p}}\right)^m \\
&\leq \left(\frac{\|1\|_p + \|(\alpha_1, \ldots, \alpha_m)\|_p}{m^{1/p}}\right)^m \\
&\leq \left(\frac{m^{1/p} + (2 + \frac{1}{\varepsilon})m^{1/p}}{m^{1/p}}\right)^m = O(\frac{1}{\varepsilon})^m,
\end{aligned}$$

   so $\Pi_{k=1}^m O(1 + \alpha_k)^d = O(1)^{md} O(\frac{1}{\varepsilon})^{md} = O(\frac{1}{\varepsilon})^{md}$.

Finally, $|\mathcal{I}_i| \leq 2^{2m} \cdot O(\frac{1}{\varepsilon})^m \cdot O(\frac{1}{\varepsilon})^{md} \leq O(\frac{1}{\varepsilon})^{m(d+1)}$. ◁

The data structure and query algorithm are similar to those we described for DFD, and the size of $\mathcal{I}_i$ and $\mathcal{I}$ is roughly the same (here there is an additional $O(\frac{1}{\varepsilon})^m$ factor in the space bound). Therefore, the query time, storage space, and preprocessing time are roughly similar, but we still need to show that the algorithm is correct.

Correctness. Consider a query curve $Q = (q_1, \ldots, q_m)$. Assume that there exists a curve $C_i \in \mathcal{C}$ such that $d_{p,2}(C_i, Q) \leq 1$. We will show that the query algorithm returns a curve C^* with $d_{p,2}(C^*, Q) \leq 1 + \varepsilon$.

Consider a point $q_k \in Q$. Denote by $q'_k \in \mathcal{G}$ the grid point closest to q_k, and let $Q' = (q'_1, \ldots, q'_m)$. We have $\|q_k - q'_k\|_2 \leq \frac{\varepsilon}{2(2m)^{1/p}}$. Let τ be an alignment such that the $\ell_{p,2}$-cost of τ w.r.t. C_i and Q is at most 1. Unlike the Fréchet distance, $\ell_{p,2}$-distance for curves does not satisfy the triangle inequality. However, by the triangle inequality under ℓ_2 and ℓ_p, we get that the $\ell_{p,2}$-cost of τ w.r.t. C_i and Q' is

$$\sigma_{p,2}(\tau(C_i,Q')) = \left(\sum_{(j,t)\in\tau}\|p_j^i - q'_t\|_2^p\right)^{1/p} \leq \left(\sum_{(j,t)\in\tau}\left(\|p_j^i - q_t\|_2 + \|q_t - q'_t\|_2\right)^p\right)^{1/p}$$
$$\leq \left(\sum_{(j,t)\in\tau}\|p_j^i - q_t\|_2^p\right)^{1/p} + \left(\sum_{(j,t)\in\tau}\|q_t - q'_t\|_2^p\right)^{1/p}$$
$$\leq 1 + \left(2m\left(\frac{\varepsilon}{2(2m)^{1/p}}\right)^p\right)^{1/p} \leq 1 + \frac{\varepsilon}{2}.$$

So $d_{p,2}(C_i,Q') \leq 1+\frac{\varepsilon}{2}$, and thus Q' is in $I_i \subseteq \mathcal{I}$. This means that $\mathcal{T}$ contains Q' with a curve $C(Q') \in \mathcal{C}$ such that $d_{p,2}(C(Q'),Q') \leq 1+\frac{\varepsilon}{2}$, and the query algorithm returns $C(Q')$. Now, again by the same argument (using an alignment with $\ell_{p,2}$-cost at most $1+\frac{\varepsilon}{2}$ w.r.t. $C(Q')$ and Q'), we get that $d_{p,2}(C(Q'),Q) \leq 1+\frac{\varepsilon}{2}+\left(2m\left(\frac{\varepsilon}{2(2m)^{1/p}}\right)^p\right)^{1/p} = 1+\varepsilon$.

We obtain the following theorem.

▶ **Theorem 14.** *There exists a data structure for the $(1+\varepsilon,r)$-ANNC under $\ell_{p,2}$-distance, with $n\cdot O(\frac{1}{\varepsilon})^{m(d+1)}$ space, $n\cdot O(\frac{1}{\varepsilon})^{m(d+1)}$ expected preprocessing time, and $O(md)$ query time.*

As mentioned in the preliminaries section, the DTW distance between two curves equals to their $\ell_{1,2}$-distance, and therefore we obtain the following theorem.

▶ **Theorem 15.** *There exists a data structure for the $(1+\varepsilon,r)$-ANNC under DTW, with $n\cdot O(\frac{1}{\varepsilon})^{m(d+1)}$ space, $n\cdot O(\frac{1}{\varepsilon})^{m(d+1)}$ expected preprocessing time, and $O(md)$ query time.*

▶ Remark 16 (Dealing with query curves and input curves of varying size). For the case of DFD, our assumption that all query curves are of length exactly k can be easily removed, by constructing k data structures $\mathcal{D}_1,\ldots,\mathcal{D}_k$, where $\mathcal{D}_i$ is our data structure constructed for query curves of length i (instead of k), for $1\leq i\leq k$. Clearly, the query time does not change. The storage space is multiplied by k, so in the case of DFD we have storage space $nk\cdot O(\frac{1}{\varepsilon})^{kd}$, but $k<2^{kd}$, so the storage space remains $n\cdot O(\frac{1}{\varepsilon})^{kd}$.

For the case of $d_{p,2}$ we can deal with queries of all sizes up to m. Our construction in Section 5 can be modified in a straightforward manner to deal with queries of size k, the space guarantee however will depend on m, upper bounded by $n\cdot O(\frac{1}{\varepsilon})^{m(d+1)}$, as in Theorem 14. From here, we can use the same approach as above.

6 A deterministic construction using a prefix tree

When implementing the dictionary $\mathcal{D}$ as a hash table, the construction of the data structure is randomized and thus in the worst case we might get higher prepeocessing time. To avoid this, we can implement $\mathcal{D}$ as a prefix tree.

6.1 Discrete Fréchet distance

In this section we describe the implementation of $\mathcal{D}$ as a prefix tree in the case of ANNC under DFD.

We can construct a prefix tree $\mathcal{T}$ for the curves in $\mathcal{I}$, where any path in $\mathcal{T}$ from the root to a leaf corresponds to a curve that is stored in it. For each $1\leq i\leq n$ and curve $\overline{Q}\in\mathcal{I}_i$, if $\overline{Q}\notin\mathcal{T}$, insert $\overline{Q}$ into $\mathcal{T}$, and set $C(\overline{Q})\leftarrow C_i$.

Each node $v \in \mathcal{T}$ corresponds to a grid point from $\mathcal{G}$. Denote the set of v's children by $N(v)$. We store with v a multilevel search tree on $N(v)$, with a level for each coordinate. The points in $\mathcal{G}$ are the grid points contained in nm balls of radius $(1+\varepsilon)r$. Thus when projecting these points to a single dimension, the number of 1-dimensional points is at most $nm \cdot \frac{\sqrt{d}(1+\varepsilon)2r}{\varepsilon r} = O(\frac{nm\sqrt{d}}{\varepsilon})$. So in each level of the search tree on $N(v)$ we have $O(\frac{nm\sqrt{d}}{\varepsilon})$ 1-dimensional points, so the query time is $O(d \log(\frac{nmd}{\varepsilon}))$.

Inserting a curve of length m to the tree $\mathcal{T}$ takes $O(md \log(\frac{nmd}{\varepsilon}))$ time. Since $\mathcal{T}$ is a compact representation of $|\mathcal{I}| = n \cdot O(\frac{1}{\varepsilon})^{dm}$ curves of length m, the number of nodes in $\mathcal{T}$ is $m \cdot |\mathcal{I}| = nm \cdot O(\frac{1}{\varepsilon})^{dm}$. Each node $v \in \mathcal{T}$ contains a search tree for its children of size $O(d \cdot |N(v)|)$, and $\sum_{v \in \mathcal{T}} |N(v)| = nm \cdot O(\frac{1}{\varepsilon})^{dm}$ so the total space complexity is $O(nmd) \cdot O(\frac{1}{\varepsilon})^{md} = n \cdot O(\frac{1}{\varepsilon})^{md}$. Constructing $\mathcal{T}$ takes $O(|\mathcal{I}| \cdot md \log(\frac{nmd}{\varepsilon})) = n \log(\frac{nmd}{\varepsilon}) \cdot O(\frac{1}{\varepsilon})^{md}$ time.

▶ **Theorem 17.** *There exists a data structure for the $(1+\varepsilon, r)$-ANNC under DFD, with $n \cdot O(\frac{1}{\varepsilon})^{dm}$ space, $n \cdot \log(\frac{n}{\varepsilon}) \cdot O(\frac{1}{\varepsilon})^{md}$ preprocessing time, and $O(md \log(\frac{nmd}{\varepsilon}))$ query time.*

Similarly, for the asymmetric case we obtain the following theorem.

▶ **Theorem 18.** *There exists a data structure for the asymmetric $(1+\varepsilon, r)$-ANNC under DFD, with $n \cdot O(\frac{1}{\varepsilon})^{dk}$ space, $nm \log(\frac{n}{\varepsilon}) \cdot \left(O(d \log m) + O(\frac{1}{\varepsilon})^{kd}\right)$ preprocessing time, and $O(kd \log(\frac{nkd}{\varepsilon}))$ query time.*

6.2 $\ell_{p,2}$-distance

For the case of ANNC under $\ell_{p,2}$-distance, the total number of curves stored in the tree $\mathcal{T}$ is roughly the same as in the case of DFD. We only need to show that for a given node v of the tree $\mathcal{T}$, the upper bound on the size and query time of the search tree associated with it are similar.

The grid points corresponding to the nodes in $N(v)$ are from n sets of m balls with radius $(1+\varepsilon)$. When projecting the grid points in one of the balls to a single dimension, the number of 1-dimensional points is at most $\frac{m^{1/p}\sqrt{d}}{\varepsilon} \cdot (1+\varepsilon)$, so the total number of projected points is at most $\frac{nm^{1+\frac{1}{p}}\sqrt{d}}{\varepsilon} \cdot (1+\varepsilon)$.

Thus in each level of the search tree of v we have $O(\frac{nm^2\sqrt{d}}{\varepsilon})$ 1-dimensional points, so the query time is $O(d \log(\frac{nmd}{\varepsilon}))$, and inserting a curve of length m into the tree $\mathcal{T}$ takes $O(md \log(\frac{nmd}{\varepsilon}))$ time. Note that the size of the search tree of v remains $O(d \cdot |N(v)|)$.

We conclude that the total space complexity is $O(\frac{nm^2\sqrt{d}}{\varepsilon}) \cdot O(\frac{1}{\varepsilon})^{m(d+1)} = n \cdot O(\frac{1}{\varepsilon})^{m(d+1)}$, constructing $\mathcal{T}$ takes $O(|\mathcal{I}| \cdot md \log(nmd/\varepsilon)) = n \log(\frac{n}{\varepsilon}) \cdot O(\frac{1}{\varepsilon})^{m(d+1)}$ time, and the total query time is $O(md \log(\frac{nmd}{\varepsilon}))$.

▶ **Theorem 19.** *There exists a data structure for the $(1+\varepsilon, r)$-ANNC under $\ell_{p,2}$-distance, with $n \cdot O(\frac{1}{\varepsilon})^{m(d+1)}$ space, $n \cdot \log(\frac{n}{\varepsilon}) \cdot O(\frac{1}{\varepsilon})^{m(d+1)}$ preprocessing time, and $O(md \log(\frac{nmd}{\varepsilon}))$ query time.*

7 Approximate range counting

In the range counting problem for curves, we are given a set $\mathcal{C}$ of n curves, each consisting of m points in d dimensions, and a distance measure for curves δ. The goal is to preprocess $\mathcal{C}$ into a data structure that given a query curve Q and a threshold value r, returns the number of curves that are within distance r from Q.

In this section we consider the following approximation version of range counting for curves, in which r is part of the input (see Remark 22). Note that by storing pointers to curves instead of just counters, we can obtain a data structure for the approximate range searching problem (at the cost of an additional $O(n)$-factor to the storage space).

▶ **Problem 20** ($(1+\varepsilon, r)$-approximate range-counting for curves)**.** *Given a parameter r and $0 < \varepsilon \leq 1$, preprocess $\mathcal{C}$ into a data structure that given a query curve Q, returns the number of all the input curves whose distance to Q is at most r plus possibly additional input curves whose distance to Q is greater than r but at most $(1+\varepsilon)r$.*

We construct the dictionary $\mathcal{D}$ (implemented as a dynamic hash table, or a prefix tree) for the curves in $\mathcal{I}$ as in Section 5, as follows. For each $1 \leq i \leq n$ and curve $\overline{Q} \in \mathcal{I}_i$, if $\overline{Q}$ is not in $\mathcal{D}$, insert it into $\mathcal{D}$ and initialize $C(\overline{Q}) \leftarrow 1$. Otherwise, if $\overline{Q}$ is in $\mathcal{D}$, update $C(\overline{Q}) \leftarrow C(\overline{Q}) + 1$. Notice that $C(\overline{Q})$ holds the number of curves from $\mathcal{C}$ that are within distance $(1+\frac{\varepsilon}{2})r$ to $\overline{Q}$. Given a query curve Q, we compute Q' as in Section 5. If Q' is in $\mathcal{D}$, we return $C(Q')$, otherwise, we return 0.

Clearly, the storage space, preprocessing time, and query time are similar to those in Section 5. We claim that the query algorithm returns the number of curves from $\mathcal{C}$ that are within distance r to Q plus possibly additional input curves whose distance to Q is greater than r but at most $(1+\varepsilon)r$. Indeed, let C_i be a curve such that $d_{dF}(C_i, Q) \leq r$. As shown in Section 5 we get $d_{p,2}(C_i, Q') \leq (1+\frac{\varepsilon}{2})r$, so Q' is in $\mathcal{I}_i$ and C_i is counted in $C(Q')$. Now let C_i be a curve such that $d_{p,2}(C_i, Q) > (1+\varepsilon)r$. If $d_{p,2}(C_i, Q') \leq (1+\frac{\varepsilon}{2})r$, then by a similar argument (switching the rolls of Q and Q') we get that $d_{p,2}(C_i, Q') \leq (1+\varepsilon)r$, a contradiction. So $d_{p,2}(C_i, Q') > (1+\frac{\varepsilon}{2})r$, and thus C_i is not counted in $C(Q')$.

We obtain the following theorem.

▶ **Theorem 21.** *There exists a data structure for the $(1+\varepsilon, r)$-approximate range-counting for curves under $\ell_{p,2}$-distance, with $n \cdot O(\frac{1}{\varepsilon})^{m(d+1)}$ space, $n \log(\frac{n}{\varepsilon}) \cdot O(\frac{1}{\varepsilon})^{m(d+1)}$ preprocessing time, and $O(md \log(\frac{nmd}{\varepsilon}))$ query time. (Under DFD, the exponent in the bounds for the space and preprocessing time is md rather than $m(d+1)$.)*

▶ Remark 22. When the threshold parameter r is part of the query, we call the problem the $(1+\varepsilon)$-*approximate range-counting problem*. Note that the reduction from $(1+\varepsilon)$-approximate nearest-neighbor to $(1+\varepsilon, r)$-approximate near-neighbor can be easily adapted to a reduction from $(1+\varepsilon)$-approximate range-counting to $(1+\varepsilon, r)$-approximate range-counting, more details will be given in a full version of this paper.

8 Simplification in d-dimensions

The algorithm of Bereg et al. [3] receives as an input a curve C consisting of m points in $\mathbb{R}^3$, and a parameter $r > 0$. In $O(m \log m)$ time, it returns a curve Π such that $d_{dF}(C, \Pi) \leq r$, and Π has the minimum number of vertices among all curves within distance r from C. The algorithm is operating in a greedy manner, by repeatedly executing Megiddo's [19] minimum enclosing ball (MEB) algorithm for points in $\mathbb{R}^3$, which takes linear time.

We generalize the algorithm of Bereg et al. for curves in $\mathbb{R}^d$, by using an algorithm presented by Kumar et al. [17] for approximated minimum enclosing ball (AMEB) in $\mathbb{R}^d$. Formally, given a set A of n points in $\mathbb{R}^d$ and a parameter $\varepsilon \in (0, 1]$, the goal is to find an enclosing ball of A with radius $r > 0$, where the minimum enclosing ball of A has radius at least $\frac{r}{1+\varepsilon}$. The algorithm of [17] can find an AMEB in $O(\frac{nd}{\varepsilon} + \varepsilon^{-4.5} \log \frac{1}{\varepsilon})$ time. In particular, given an additional parameter $r > 0$, this algorithm either returns an enclosing ball of A with radius $(1+\varepsilon)r$, or declares that the minimum enclosing ball of A has radius larger than r.

Next, we describe our modified algorithm. Consider a curve $C = (x_1, \ldots, x_m)$, and denote $C[i,j] = (x_i, \ldots, x_j)$. The following sub-procedure takes as an input a curve A and returns a point y and an index s, such that the ball with radius $(1+\varepsilon)r$ centered at y covers the prefix $A[1,s]$, and (if $s < |A|$) the minimum enclosing ball of $A[1,s+1]$ has radius larger than r.

1. By iterative probing, using an algorithm for AMEB, find some t such that $A[1,2^t]$ can be covered by a ball of radius $(1+\varepsilon)r$, while $A[1,2^{t+1}]$ cannot be covered by a ball of radius r. If all the points in A can be enclosed by a single ball of radius $(1+\varepsilon)r$ centered at y, simply return y and $|A|$.
2. By binary search, again using an algorithm for AMEB, find some $s \in [2^t, 2^{t+1})$ such that $A[1,s]$ can be covered by a ball of radius $(1+\varepsilon)r$, and $A[1,s+1]$ cannot be covered by a ball of radius r. Let $y \in \mathbb{R}^d$ be the center of this ball. Return y and s.

Starting from the input $A = C[1,m]$, repeat the above sub-procedure such that in each step the input is the suffix of C that was not yet covered by the previous steps (i.e. $A[s+1,m]$). Let $(y_1, \ldots, y_q)$ be the sequence of output points.

Lemma 10 is an easy corollary of the following lemma.

▶ **Lemma 23.** *Let C be a curve consisting of m points in $\mathbb{R}^d$. Given parameters $r > 0$, and $\varepsilon \in (0,1]$, the algorithm above runs in $O\left(\frac{d \cdot m \log m}{\varepsilon} + m \cdot \varepsilon^{-4.5} \log \frac{1}{\varepsilon}\right)$ time and returns a curve $\Pi = (y_1, \ldots, y_q)$ such that $d_{dF}(C, \Pi) \leq (1+\varepsilon)r$. Furthermore, for every curve Π' with less than q points, it holds that $d_{dF}(C, \Pi') > r$.*

Proof sketch. We start by analyzing the running time for a single iteration of the sub-procedure, when using the algorithm of [17] to find an AMEB. The total time for the first step of the sub-procedure (finding t) is

$$\sum_{i=1}^{t+1} O(\frac{2^i \cdot d}{\varepsilon} + \varepsilon^{-4.5} \log \frac{1}{\varepsilon}) = O(\frac{2^t \cdot d}{\varepsilon} + t \cdot \varepsilon^{-4.5} \log \frac{1}{\varepsilon}).$$

In the second step, there are $O(t)$ executions of [17] on a set of size at most 2^{t+1}, so the total time for this step is $t \cdot O(\frac{2^t \cdot d}{\varepsilon} + \varepsilon^{-4.5} \log \frac{1}{\varepsilon})$.

Let m_i be the length of the subcurve covered by the point y_i that was found in the i'th iteration of the sub-procedure. The total time spent for finding y_i is therefore $\log m_i \cdot O(\frac{m_i \cdot d}{\varepsilon} + \varepsilon^{-4.5} \log \frac{1}{\varepsilon})$, and the total running time of the algorithm is

$$\sum_{i=1}^{q} \log m_i \cdot O\left(\frac{m_i \cdot d}{\varepsilon} + \varepsilon^{-4.5} \log \frac{1}{\varepsilon}\right) = O\left(\frac{d \cdot m \log m}{\varepsilon} + m \cdot \varepsilon^{-4.5} \log \frac{1}{\varepsilon}\right) ,$$

where we used the the concavity of the log function, and the fact $\sum_{i=1}^{q} m_i = m$.

Next we argue the correctness. Clearly, $d_{dF}(C, \Pi) \leq (1+\varepsilon)r$. Let $s_0 = 0, s_1, \ldots, s_q = m$ be the sequence of indices (of vertices in C) found during the execution of the algorithm, such that the ball of radius $(1+\varepsilon)r$ around y_i covers $C[s_{i-1}+1, s_i]$. It follows by a straightforward induction that every curve Π' with less that i points will be at distance greater than r from $C[1, s_{i-1}+1]$. The lemma now follows. ◀

9 Remark on dimension reduction

In general, when the dimension d is large, i.e. $d \gg \log(nm)$, one can use dimension reduction (using the celebrated Johnson-Lindenstrauss lemma [16]) in order to achieve a better running time, at the cost of inserting randomness in the prepossessing and query procedure. However,

such an approach can work only against an oblivious adversary, as it will necessarily fail for some curves. Recently Narayanan and Nelson [20] (improving [11, 18]) proved a terminal version of the JL-lemma. Given a set K of k points in $\mathbb{R}^d$ and $\varepsilon \in (0,1)$, there is a dimension reduction function $f : \mathbb{R}^d \to \mathbb{R}^{O(\frac{\log k}{\varepsilon^2})}$ such that for every $x \in K$ and $y \in \mathbb{R}^d$ it holds that $\|x-y\|_2 \leq \|f(x)-f(y)\|_2 \leq (1+\varepsilon)\cdot\|x-y\|_2$.

This version of dimension reduction can be used such that the query remains deterministic and always succeeds. The idea is to take all the nm points from all the input curves to be the terminals, and let f be the terminal dimension reduction. We transform each input curve $P = (p_1, \ldots, p_m)$ into $f(P) = (f(p_1), \ldots, f(p_m))$, a curve in $\mathbb{R}^{O(\frac{\log nm}{\varepsilon^2})}$. Given a query $Q = (q_1, \ldots, q_m)$ we transform it to $f(Q) = (f(q_1), \ldots, f(q_m))$. Since the pairwise distances between every query point to all input points are preserved, so is the distance between the curves. Specifically, the $d_{p,2}$ distance w.r.t. any alignment τ is preserved up to a $1+\varepsilon$ factor, and therefore we can reliably use the answer received using the transformed curves.

References

1 Peyman Afshani and Anne Driemel. On the complexity of range searching among curves. In *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2018, New Orleans, LA, USA, January 7-10, 2018*, pages 898–917, 2018. doi:10.1137/1.9781611975031.58.

2 Boris Aronov, Omrit Filtser, Michael Horton, Matthew J. Katz, and Khadijeh Sheikhan. Efficient nearest-neighbor query and clustering of planar curves. In *Algorithms and Data Structures - 16th International Symposium, WADS 2019, Edmonton, AB, Canada, August 5-7, 2019, Proceedings*, pages 28–42, 2019. doi:10.1007/978-3-030-24766-9_3.

3 Sergey Bereg, Minghui Jiang, Wencheng Wang, Boting Yang, and Binhai Zhu. Simplifying 3d polygonal chains under the discrete Fréchet distance. In *LATIN 2008: Theoretical Informatics, 8th Latin American Symposium, Búzios, Brazil, April 7-11, 2008, Proceedings*, pages 630–641, 2008. doi:10.1007/978-3-540-78773-0_54.

4 Karl Bringmann. Why walking the dog takes time: Fréchet distance has no strongly subquadratic algorithms unless SETH fails. In *55th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2014, Philadelphia, PA, USA, October 18-21, 2014*, pages 661–670, 2014. doi:10.1109/FOCS.2014.76.

5 Kevin Buchin, Anne Driemel, Joachim Gudmundsson, Michael Horton, Irina Kostitsyna, Maarten Löffler, and Martijn Struijs. Approximating (k, l)-center clustering for curves. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2019, San Diego, California, USA, January 6-9, 2019*, pages 2922–2938, 2019. doi:10.1137/1.9781611975482.181.

6 Mark de Berg, Atlas F. Cook IV, and Joachim Gudmundsson. Fast Fréchet queries. *Comput. Geom.*, 46(6):747–755, 2013. doi:10.1016/j.comgeo.2012.11.006.

7 Mark de Berg, Joachim Gudmundsson, and Ali D. Mehrabi. A dynamic data structure for approximate proximity queries in trajectory data. In *Proceedings of the 25th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems, GIS 2017, Redondo Beach, CA, USA, November 7-10, 2017*, pages 48:1–48:4, 2017. doi:10.1145/3139958.3140023.

8 Anne Driemel and Sariel Har-Peled. Jaywalking your dog: Computing the Fréchet distance with shortcuts. *SIAM J. Comput.*, 42(5):1830–1866, 2013. doi:10.1137/120865112.

9 Anne Driemel, Ioannis Psarros, and Melanie Schmidt. Sublinear data structures for short Fréchet queries. *CoRR*, abs/1907.04420, 2019. arXiv:1907.04420.

10 Anne Driemel and Francesco Silvestri. Locality-Sensitive Hashing of Curves. In *Proceedings of the 33rd International Symposium on Computational Geometry*, volume 77, pages 37:1–

37:16, Brisbane, Australia, July 2017. Schloss Dagstuhl–Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.SoCG.2017.37.

11 Michael Elkin, Arnold Filtser, and Ofer Neiman. Terminal embeddings. *Theor. Comput. Sci.*, 697:1–36, 2017. doi:10.1016/j.tcs.2017.06.021.

12 Ioannis Z. Emiris and Ioannis Psarros. Products of Euclidean metrics and applications to proximity questions among curves. In *34th International Symposium on Computational Geometry, SoCG 2018, June 11-14, 2018, Budapest, Hungary*, pages 37:1–37:13, 2018. doi:10.4230/LIPIcs.SoCG.2018.37.

13 Sariel Har-Peled, Piotr Indyk, and Rajeev Motwani. Approximate nearest neighbor: Towards removing the curse of dimensionality. *Theory of Computing*, 8(1):321–350, 2012. doi:10.4086/toc.2012.v008a014.

14 Piotr Indyk. *High-dimensional computational geometry*. PhD thesis, Stanford University, 2000. see here.

15 Piotr Indyk. Approximate nearest neighbor algorithms for Fréchet distance via product metrics. In *Proceedings of the 8th Symposium on Computational Geometry*, pages 102–106, Barcelona, Spain, June 2002. ACM Press. doi:10.1145/513400.513414.

16 William Johnson and Joram Lindenstrauss. Extensions of Lipschitz mappings into a Hilbert space. *Contemporary Mathematics*, 26:189–206, 1984. doi:10.1090/conm/026/737400.

17 Piyush Kumar, Joseph S. B. Mitchell, and E. Alper Yildirim. Comuting core-sets and approximate smallest enclosing hyperspheres in high dimensions. In *Proceedings of the Fifth Workshop on Algorithm Engineering and Experiments, Baltimore, MD, USA, January 11, 2003*, pages 45–55, 2003.

18 Sepideh Mahabadi, Konstantin Makarychev, Yury Makarychev, and Ilya P. Razenshteyn. Nonlinear dimension reduction via outer bi-Lipschitz extensions. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2018, Los Angeles, CA, USA, June 25-29, 2018*, pages 1088–1101, 2018. doi:10.1145/3188745.3188828.

19 Nimrod Megiddo. Linear programming in linear time when the dimension is fixed. *J. ACM*, 31(1):114–127, 1984. doi:10.1145/2422.322418.

20 Shyam Narayanan and Jelani Nelson. Optimal terminal dimensionality reduction in Euclidean space. In *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing, STOC 2019, Phoenix, AZ, USA, June 23-26, 2019*, pages 1064–1069, 2019. doi:10.1145/3313276.3316307.

21 Rasmus Pagh and Flemming Friche Rodler. Cuckoo hashing. *Journal of Algorithms*, 51(2):122–144, 2004. doi:10.1016/j.jalgor.2003.12.002.

22 Gregory Shakhnarovich, Trevor Darrell, and Piotr Indyk. *Nearest-neighbor methods in learning and vision: theory and practice (neural information processing)*. The MIT press, 2006. see here.