Approximating the Fréchet Distance When Only One Curve Is c-Packed

Joachim Gudmundsson
School of Computer Science, University of Sydney, Australia

Tiancheng Mai
School of Computer Science, University of Sydney, Australia

Sampson Wong
Department of Computer Science, University of Copenhagen, Denmark

Abstract

One approach to studying the Fréchet distance is to consider curves that satisfy realistic assumptions. By now, the most popular realistic assumption for curves is c-packedness. Existing algorithms for computing the Fréchet distance between c-packed curves require both curves to be c-packed. In this paper, we only require one of the two curves to be c-packed. Our result is a nearly-linear time algorithm that $(1+\varepsilon)$-approximates the Fréchet distance between a c-packed curve and a general curve in $\mathbb{R}^d$, for constant values of ε, d and c.

2012 ACM Subject Classification Theory of computation → Computational geometry

Keywords and phrases Fréchet distance, c-packed curve, approximation algorithm

Digital Object Identifier 10.4230/LIPIcs.ISAAC.2024.37

Related Version *Full Version*: https://arxiv.org/abs/2407.05114 [15]

1 Introduction

The Fréchet distance [13] is a popular similarity measure between curves. The Fréchet distance has a variety of applications, from geographic information science [23, 24, 26] to computational biology [21, 29] and data mining [22, 28]. The Fréchet distance can be seen as the minimum leash length of a dog walking problem.

Suppose a person and a dog walk along two polygonal curves P and Q, respectively. The goal of both the person and the dog is to walk along the path, independently and at possibly different speeds, but without leaving the path or going backwards. The leash length of a given walk is defined to be the maximum distance attained between the person and the dog. The Fréchet distance is the globally minimum leash length over all possible walks.

The Fréchet distance can be computed between a pair of polygonal curves in nearly-quadratic time. Alt and Godau [1] provided an $O(n^2 \log n)$ time exact algorithm for computing the Fréchet distance. Buchin, Buchin, Meulemans and Mulzer [6] provided a randomised exact algorithm that computes the Fréchet distance in time $O(n^2\sqrt{\log n}(\log\log n)^{3/2})$ on a pointer machine, and in time $O(n^2(\log\log n)^2))$ on word RAM.

Conditional lower bounds imply that the Fréchet distance problem is unlikely to admit a strongly subquadratic time algorithm. Bringmann [2] showed that, under the Strong Exponential Time Hypothesis, the Fréchet distance cannot be computed in time $O(n^{2-\delta})$ for any $\delta > 0$, if we allow for approximation factors up to 1.001. Buchin, Ophelders and Speckmann [7] showed the same conditional lower bound even if we allow for approximation factors up to 3, and even if the curves are one dimensional.

One approach to circumvent the conditional lower bounds on the Fréchet distance is to focus on curves that satisfy realistic assumptions. Realistic assumptions reflect the spatial distribution of curves from real-world data sets [17]. The most popular realistic input assumption for curves under the Fréchet distance is c-packedness [12]. A curve $\pi \in \mathbb{R}^d$ is c-packed if for all $r > 0$, the total length of π inside any ball of radius r is upper bounded by cr.

Driemel, Har-Peled and Wenk [12] introduced the c-packedness assumption and presented a $(1 + \varepsilon)$-approximation algorithm for the Fréchet distance between a pair of c-packed curves. Their algorithm runs in $O(cn/\varepsilon + cn \log n)$ time for curves in $\mathbb{R}^d$. Bringmann and Künnemann [3] improved the running time of the algorithm to $O(\frac{cn}{\sqrt{\varepsilon}} \log^2(1/\varepsilon) + cn \log n)$ for curves in $\mathbb{R}^d$. Assuming the Strong Exponential Time Hypothesis, Bringmann [2] showed that (i) for sufficiently small constants $\varepsilon > 0$ there is no $(1 + \varepsilon)$-approximation in time $O((cn)^{1-\delta})$ for any $\delta > 0$, and (ii) in any dimension $d \geq 5$ there is no $(1 + \varepsilon)$-approximation in time $O((cn/\sqrt{\varepsilon})^{1-\delta})$ for any $\delta > 0$.

Existing algorithms [3, 12] for computing the Fréchet distance between c-packed curves require that both curves are c-packed. An open problem is whether the Fréchet distance can be approximated efficiently when only one curve is c-packed. This asymmetric case may occur if the two curves come from two different data sets. For example, in error detection we may want to match a curve containing errors to a curve close to the ground truth.

▶ **Problem 1.** *Can the Fréchet distance be approximated efficiently if only one of the two curves is c-packed? In particular, for constant values of ε, d and c, can we obtain a subquadratic time $(1+\varepsilon)$-approximation of the Fréchet distance between a c-packed curve and a general curve in $\mathbb{R}^d$?*

We resolve Problem 1 in the affirmative. Our result is an $O(c^3(n + m) \log^{2d+1}(n) \log m)$ time algorithm that $(1 + \varepsilon)$-approximates the Fréchet distance between a c-packed curve with n vertices in $\mathbb{R}^d$ and a general curve with m vertices in $\mathbb{R}^d$, where ε is a constant. In other words, to $(1 + \varepsilon)$-approximate the Fréchet distance in nearly-linear time, our result implies that it suffices to assume that only one of the two curves is c-packed. Our result is stated formally in Theorem 11. Note that for constant values of d, the running time is also polynomial in c and ε.

1.1 Related work

By now, the most popular realistic input assumption for curves under the Fréchet distance is c-packedness. The c-packedness assumption has been applied to a wide variety of Fréchet distance problems. Typically, these algorithms incur an approximation factor of $(1 + \varepsilon)$, and have a polynomial dependence on ε^{-1}. Chen, Driemel, Guibas, Nguyen and Wenk [8] study the map matching problem between a c-packed curve and realistic graph, that is, to compute a path in the graph that is most similar to the c-packed curve. Har-Peled and Raichel [19] compute the mean curve of a set of c-packed curves. The mean curve is a curve that minimises its maximum weak Fréchet distance to the set of curves. Driemel and Har-Peled [11] consider a variant of the Fréchet distance on c-packed curves, where any subcurve of the c-packed curve can be replaced by a shortcut segment. Brüning, Conradi and Driemel [4] and Gudmundsson, Huang, van Renssen and Wong [14] study two distinct variants of the subtrajectory clustering problem on c-packed curves, that is, to detect trajectory patterns by computing clusters of subcurves. Van der Hoog, Rotenberg and Wong [27] study data structures for c-packed curves under the discrete Fréchet distance. Conradi, Driemel and Kolbe [9] consider the approximate nearest neighbour problem for c-packed curves in doubling metrics. Conradi, Driemel and Kolbe [10] compute the Fréchet distance between c-packed piecewise continuous smooth curves.

Given a polygonal curve, the problem of computing its packedness value c has been considered. Gudmundsson, Sha and Wong [17] provide a 6.001-approximation algorithm that runs in $O(n^{4/3}\log^9 n)$ time for curves in $\mathbb{R}^2$. They also provided an implementation for a 2-approximation algorithm that runs in $O(n^2)$, and verified that $c < 50$ for a majority of data sets that were tested. Har-Peled and Zhou [20] provide a randomised 288.001-approximation algorithm that runs in $O(n\log^2 n)$ time and succeeds with high probability.

The c-packedness assumption can be applied to any set of edges, as a result, c-packed graphs have also been studied. Gudmundsson and Smid [18] study the map matching problem between a curve with long edges and a c-packed graph with long edges. They consider the data structure variant, where the graph is known in preprocessing time and the curve is only known at query time. Gudmundsson, Seybold and Wong [16] generalise the result of [18] and provide a map matching data structure for any c-packed graph and for any query curve.

1.2 Notation

Let $\varepsilon > 0$ be a positive real number. Without loss of generality, we can assume $0 < \varepsilon < \frac{1}{2}$, as providing a $(1+\varepsilon)$-approximation for smaller values of ε also provides a $(1+\varepsilon)$-approximation for larger values of ε.

Let d be a fixed positive integer, and let $\mathbb{R}^d$ be d-dimensional Euclidean space. A polygonal curve $P = p_1 \dots p_n$ in $\mathbb{R}^d$ consists of n vertices $\{p_i\}_{i=1}^n$ connected by $n-1$ straight line segments $\{p_ip_{i+1}\}_{i=1}^{n-1}$, where $p_i \in \mathbb{R}^d$ and $p_ip_{i+1} \subset \mathbb{R}^d$.

We define c-packedness. Let c be a positive real number. A polygonal curve P in $\mathbb{R}^d$ is c-packed if, for any radius $r > 0$ and for any ball $B(p,r)$ centred at $p \in \mathbb{R}^d$ with radius r, the set of segments in $P \cap B(p,r)$ has total length upper bounded by cr.

Next, we define the Fréchet distance. Let $P = p_1 \dots p_n$. With slight abuse of notation, define the function $P : [1,n] \to \mathbb{R}^d$ so that $P(i) = p_i$ for all integers $i \in \{1,\dots,n\}$, and $P(i+x) = (1-x)p_i + xp_{i+1}$ for all reals $x \in [0,1]$. Let $\Gamma(n)$ be the space of all continuous, non-decreasing, surjective functions from $[0,1] \to [1,n]$. For a pair of polygonal curves $P = p_1,\dots,p_n$ and $Q = q_1,\dots,q_m$, we define the Fréchet distance to be

$$d_F(P,Q) = \inf_{\substack{\alpha\in\Gamma(n)\\ \beta\in\Gamma(m)}} \max_{\mu\in[0,1]} d(P(\alpha(\mu)), Q(\beta(\mu)))$$

where $d(\cdot,\cdot)$ denotes the Euclidean distance in $\mathbb{R}^d$.

2 Decision algorithm

In this section, we solve the decision version of the Fréchet distance problem. We defer the optimisation version of the Fréchet distance problem to Section 3. Both the decision and optimisation versions will incur an approximation factor of $(1+\varepsilon)$.

We formally define the decision version. First we define the exact decision version. Let r be a positive real number. Let $P = p_1p_2 \dots p_n$ be a c-packed curve in $\mathbb{R}^d$ and let $Q = q_1q_2 \dots q_m$ be a general curve in $\mathbb{R}^d$. Given P, Q and r, the exact decision problem is to answer whether (i) $d_F(P,Q) \le r$ or (ii) $d_F(P,Q) > r$, where $d_F(\cdot,\cdot)$ denotes the Fréchet distance. Unfortunately, in our case, we will not be able to decide between (i) and (ii) exactly. Therefore, we instead solve the approximate decision version. In the approximate decision problem, we are additionally allowed a third option, that is (iii) to provide a $(1+\varepsilon)$-approximation for $d_F(P,Q)$.

We will build a decider for the approximate decision version, for any fixed $0 < \varepsilon < \frac{1}{2}$. Given any P, Q and r, the decider returns either (i), (ii) or (iii). The decider requires the c-packed curve to be simplified. We will first describe the simplification procedure (Section 2.1), then we will construct the fuzzy decider (Section 2.2), and finally we will combine two fuzzy deciders into a complete approximate decider (Section 2.3).

2.1 Simplification

The first step in the decision algorithm is to simplify the c-packed curve P. We will use the simplification algorithm in Driemel, Har-Peled and Wenk [12].

▶ **Fact 2** ([12]). *Given $\mu > 0$ and a polygonal curve $\pi = p_1 p_2 p_3 \ldots p_k$ in $\mathbb{R}^d$, we can compute in $O(k)$ time a simplification $\text{simpl}(\pi, \mu)$ with the following properties:*

a) *for any vertex $p \in \pi$ there exists a vertex $q \in \text{simpl}(\pi, \mu)$ such that $d(p, q) \le \mu$,*
b) *$d_F(\pi, \text{simpl}(\pi, \mu)) \le \mu$,*
c) *all segments in $\text{simpl}(\pi, \mu)$ have length at least μ (except the last),*
d) *if π is c-packed, then $\text{simpl}(\pi, \mu)$ is $6c$-packed.*

Proof. We state Algorithm 2.1 from [12], since we will use it in Section 3.1 to determine the critical values of our algorithm. Mark the initial vertex p_1 and set it as the current vertex. Scan the polygonal curve from the current vertex until it reaches the first vertex p_i that is at least μ away from the current vertex. Mark p_i and set it as the current vertex. Repeat this until the final vertex, and mark the final vertex. Set the marked vertices to be the simplified curve, and denote it as $\text{simpl}(\pi, \mu)$. See Figure 1. Fact 2a follows from Algorithm 2.1 in [12]. Facts 2b, 2c and 2d follow from Lemma 2.3, Remark 2.2 and Lemma 4.3 in [12]. ◀

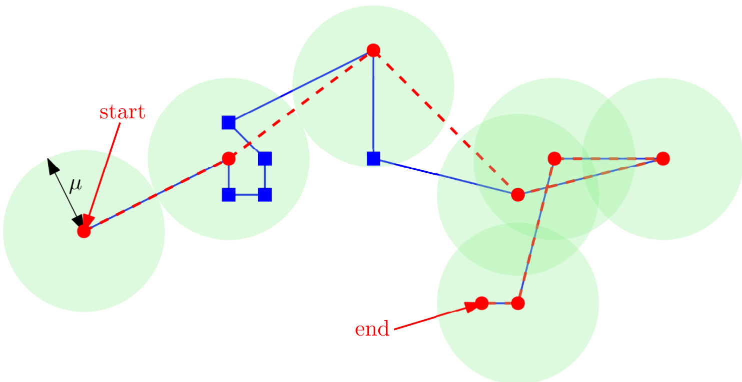

Figure 1 A polygonal trajectory P (blue) and its μ-simplification (red dashed). The vertices marked with blue squares are on P but not included in the simplification.

2.2 Fuzzy decider

The second step in the decision algorithm is to construct a fuzzy decider. Let $\varepsilon' = \varepsilon/30$. Given P, Q and r, the fuzzy decision problem is to answer whether (i) $d_F(P, Q) \le (1+\varepsilon'/2)r$, or (ii) $d_F(P, Q) > (1 - 2\varepsilon')r$. We call the decision problem fuzzy as there is a fuzzy region $((1 - 2\varepsilon')\, r, (1 + \varepsilon'/2)\, r]$ where it would be acceptable to return either (i) or (ii). Note that unlike the complete approximate decider, for the fuzzy decider, there is no option (iii).

The overall approach in the fuzzy decider is to approximate the optimal walks along K and Q, where K is the simplification of P from Fact 2. In particular, our approach is to guess how far along K we are when we reach vertex q_i on Q. We use a layered directed graph to model the walk along K, where each layer corresponds to the walk reaching q_i on Q.

The fuzzy decision algorithm constructs a layered directed graph and searches it for a suitable walk. We divide the fuzzy decision algorithm into three steps. The first step is to query a range searching data structure [25] to construct the vertices of the graph (Section 2.2.1). The second step is to query an approximate Fréchet distance data structure [11] to construct the directed edges of the graph (Section 2.2.2). The third step is to run a breadth first search and then to return either (i) or (ii) (Section 2.2.3).

2.2.1 Constructing the vertices

The first step in the fuzzy decider is to construct the vertices of the layered directed graph. Let $\delta = \varepsilon'/2 = \varepsilon/60$. Construct the simplification $K = \mathrm{simpl}(P, \delta r)$ using Fact 2. Recall that layer i corresponds to candidate positions on K when we reach q_i on Q. Formally, define layer i to be $W_i = \{w_{i,j}\}$. All points $w_{i,j} \in W_i$ satisfy $w_{i,j} \in K$ and $d(w_{i,j}, q_i) \le 2r$. Note that $w_{i,j}$ is not necessarily a vertex of P, but rather a point on an edge of $K = \mathrm{simpl}(P, \delta r)$. To construct W_i, we require the data structure of Schwarzkopf and Vleugels [25], which is a range searching data structure for low density environments.

▶ **Definition 3.** *A set of objects Σ is k-low density if, for every box H, there are at most k objects in H that intersect it and are larger than it. The size of an object is the size of its smallest enclosing box.*

▶ **Fact 4** (Theorem 3 in [25]). *A k-low density environment Σ of n objects in $\mathbb{R}^d$ can be stored in a data structure of size $O(n \log^{d-1} n + kn)$, such that it takes $O(\log^{d-1} n + k)$ time to report all $E \in \Sigma$ that contains a given query point $q \in \mathbb{R}^d$. The data structure can be computed in $O(n \log^d n + kn \log n)$ time.*

We apply Fact 4 to the curve K. In particular, we turn K into a low-environment in $\mathbb{R}^{d+1}$ by using the trough construction of Gudmundsson, Seybold and Wong [16]. The same trough construction was also used in [5].

▶ **Lemma 5.** *Let $\delta > 0$ be fixed. Let P be a c-packed curve with n vertices in $\mathbb{R}^d$. Let $K = \mathrm{simpl}(P, \delta r)$. We can preprocess K into a data structure of $O(n \log^d n + c\delta^{-1} n)$ size, so that given a query point $q \in \mathbb{R}^d$, the data structure can return in $O(\log^d n + c\delta^{-1})$ time all $O(c\delta^{-1})$ edges of K that are within a distance of $2r$ from q. The preprocessing time is $O(n \log^{d+1} n + c\delta^{-1} n \log n)$.*

Proof. The curve K is $6c$-packed by Fact 2d. Next, we generalise the trough construction of Gudmundsson, Seybold and Wong [16] to $(d+1)$-dimensions. We define a trough object in $\mathbb{R}^{d+1}$ for every segment $e \in K$ by $\mathrm{trough}(e, \delta) = \{(x_1, \ldots, x_d, z) : d\left((x_1, \ldots, x_d), e\right) \le 4z \le 8\delta^{-1}|e|\}$, where $d(\cdot, \cdot)$ and $|\cdot|$ are measured under the Euclidean metric in $\mathbb{R}^d$. Let T be the set of all trough objects. By Lemma 23 in [17], T is an $O(c\delta^{-1})$-low-density environment. We apply the data structure from Fact 4 on the environment T.

Given a query point $q = (x_1, \ldots, x_d)$, we query the data structure for all troughs that contain the $(d+1)$-dimensional point $(x_1, \ldots, x_d, r/2)$. Suppose the data structure returns a set of k objects $\{\mathrm{trough}(e_i, \delta)\}_{i=1}^k$. Then $k = O(c\delta^{-1})$, since T is an $O(c\delta^{-1})$-low-density environment, and q has zero size. From the set of k troughs we extract the set of k edges $\{e_i\}_{i=1}^k$.

The running times follow from Fact 4 and from T being an $O(c\delta^{-1})$-low-density environment. It remains to prove the correctness of the query. Recall the definition of the trough that $(x_1, \ldots, x_d, r/2) \in \text{trough}(e, \delta)$ if and only if $d((x_1, \ldots, x_d), e) \leq 4 \cdot \frac{r}{2} \leq 8\delta^{-1}|e|$. In particular, $d(q, e) \leq 2r$ covers all edges in K that intersect a ball of radius $2r$ centred at q and $4\delta^{-1}|e| \geq r$ covers all edges of length at least $\delta r/4$. Since K is also (δr)-simplified, all edges of K (except for the last edge) are at least of length δr by Fact 2c. We can check the last edge of K separately. ◀

We use Lemma 5 to construct W_i for all $1 \leq i \leq m$. Recall that $Q = q_1 \ldots q_m$. Query the data structure in Lemma 5 to obtain all edges in $K = \text{simpl}(P, \delta r)$ that are within a distance of $2r$ from q_i. Let this set of edges be T_i. Note that $|T_i| = O(c\delta^{-1})$, since K is $6c$-packed and each edge in T_i has length at least δr. For each edge $e_{i,j} \in T_i$, we choose $O(\delta^{-1})$ evenly spaced points on the chord $e_{i,j} \cap B(q_i, 2r)$, so that the distance between two consecutive points on the chord is less than δr. We add these evenly spaced points to W_i for each $e_{i,j} \in T_i$, so that in total, $|W_i| = O(c\delta^{-2})$. See Figure 2.

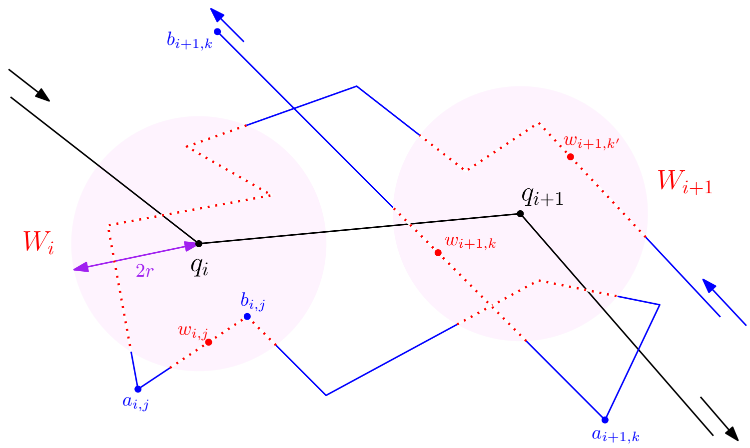

Figure 2 The general curve Q (black), the (δr)-simplification K (blue) and two candidate sets W_i and W_{i+1} (red dots). The coloured arrows indicate the order of vertices on the curve. A candidate set W_i (red dots) contains evenly spaced points on K chords that are at most a distance $2r$ away from q_i, i.e., in the violet shading. The point $w_{i,j}$ is on the edge $a_{i,j}b_{i,j}$ and the point $w_{i+1,k}$ is on the edge $a_{i+1,k}b_{i+1,k}$.

This completes the construction of W_i for $1 \leq i \leq m$. Since q_1, q_m must be matched to p_1, p_n respectively, we can simplify the sets $W_1 = \{p_1\}$ and $W_m = \{p_n\}$. The vertices of our graph are $\cup_{i=1}^{m} W_i$, which completes the first step of the construction of the fuzzy decider.

2.2.2 Constructing the edges

The second step in the fuzzy decider is to construct the edges of the layered directed graph. Each edge in the graph is a directed edge from W_i to W_{i+1} for some $1 \leq i \leq m-1$. A directed edge from $w_{i,j} \in W_i$ to $w_{i+1,k} \in W_{i+1}$ models a simultaneous walk, from $w_{i,j}$ to $w_{i+1,k}$ and from q_i to q_{i+1}, on K and Q respectively. We only add this directed edge into the graph if its associated walk is feasible. To decide whether the walk is feasible, we check two conditions. The first condition is that $w_{i,j}$ preceeds $w_{i+1,k}$ along the curve K. The second condition is whether the Fréchet distance between the subcurve $K\langle w_{i,j}, w_{i+1,k}\rangle$ and the segment $q_i q_{i+1}$ is at most r. See Figure 3. To efficiently check the second condition, we require the approximate Fréchet distance data structure of Driemel and Har-Peled [11].

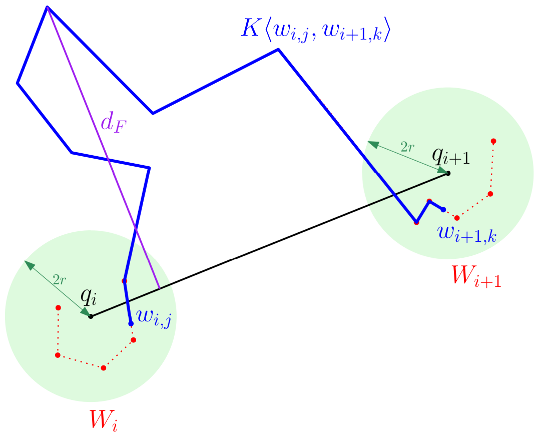

Figure 3 The Fréchet distance (purple), between a segment (q_i, q_{i+1}) (black) and subcurve $K\langle w_{i,j}, w_{i+1,k}\rangle$ (blue). A candidate set W_i (red dots) contains evenly spaced points on K chords that are at most a distance $2r$ away from q_i, i.e., in the green shading.

▶ **Fact 6** (Theorem 5.9 in [11]). *Given $\delta > 0$ and a polygonal curve Z with n vertices in $\mathbb{R}^d$, one can construct a data structure in $O(\delta^{-2d} \log^2(1/\delta) n \log^2 n)$ time that uses $O(\delta^{-2d} \log^2(1/\delta) n)$ space, such that for a query segment pq, and any two points u and v on the curve, one can $(1+\delta)$-approximate the distance $d_F(Z\langle u, v\rangle, pq)$ in $O(\delta^{-2} \log n \log\log n)$ query time.*

We construct the data structure in Fact 6 on the curve K. Let $1 \le i \le m-1$, $w_{i,j} \in W_i$ and $w_{i+1,k} \in W_{i+1}$. We query the data structure in Fact 6 to compute a $(1+\delta)$-approximation of $d_F(K\langle w_{i,j}, w_{i+1,k}\rangle, q_i q_{i+1})$. If the reported value is at most r, then we insert the directed edge from $w_{i,j}$ to $w_{i+1,k}$. We repeat this for all $1 \le i \le m-1$, $w_{i,j} \in W_i$ and $w_{i+1,k} \in W_{i+1}$. This completes the construction of the edges in the directed graph, and completes the second step of the fuzzy decider.

2.2.3 Returning either (i) or (ii)

The third step of the fuzzy decider is to run a breadth first search on the layered directed graph. Recall that $W_1 = \{p_1\}$ and $W_m = \{p_n\}$. We use the breadth first search to decide whether there is a directed path from p_1 to p_n. Recall that $\varepsilon' = \varepsilon/30$ and $\delta = \varepsilon/60$. If there is a directed path, we return (i) $d_F(P, Q) \le (1 + \varepsilon'/2)r$. Otherwise, we return (ii) $d_F(P, Q) > (1 - 2\varepsilon')r$. Next, we prove the correctness of the fuzzy decider. We have two cases.

- There is a directed path from p_1 to p_n in the layered directed graph. Let the directed path be $c_1 \dots c_m$. Then $c_i \in W_i$ for all $1 \le i \le m$. We match the vertex q_i to c_i for all $1 \le i \le m$. We match the segment $q_i q_{i+1}$ to the subcurve $K\langle c_i, c_{i+1}\rangle$ for all $1 \le i \le m-1$. Since there is a directed edge from c_i to c_{i+1}, we have that the estimated Fréchet distance between the segment $q_i q_{i+1}$ and the subcurve $d_F(K\langle c_i, c_{i+1}\rangle, q_i q_{i+1})$ is at most r. Formally, we have $C_i \le r$, where

  $$d_F(K\langle c_i, c_{i+1}\rangle, q_i q_{i+1}) \le C_i \le (1+\delta) \cdot d_F(K\langle c_i, c_{i+1}\rangle, q_i q_{i+1}).$$

  In particular, we have $d_F(K\langle c_i, c_{i+1}\rangle, q_i q_{i+1}) \le r$. Taking the maximum over all $1 \le i \le m-1$, we get

  $$d_F(K, Q) \le \max_{i=1,\dots,m-1} d_F(K\langle c_i, c_{i+1}\rangle, q_i q_{i+1}) \le r.$$

By Fact 2b, we have $d_F(P, K) \leq \delta r$. Since the Fréchet distance obeys the triangle inequality, we have

$$d_F(P, Q) \leq d_F(P, K) + d_F(K, Q) \leq r + \delta r \leq (1 + \varepsilon'/2)r.$$

Therefore, it is correct to return (i) $d_F(P, Q) \leq (1 + \varepsilon'/2)r$ in the case where there is a directed path from p_1 to p_n.

- There is no directed path from p_1 to p_n in the layered directed graph. Let $r^* = d_F(P, Q)$. Suppose that for the optimal Fréchet distance between P and Q, we match $q_i \in Q$ to $p_i^* \in P$ for all $1 \leq i \leq m$. Therefore $d(q_i, p_i^*) \leq r^*$. Let $r' = d_F(K, Q)$. Suppose that for the optimal Fréchet distance between K and Q, we match $q_i \in Q$ to $k_i^* \in K$ for all $1 \leq i \leq m$. Therefore $d(q_i, k_i^*) \leq r'$. Since the Fréchet distance obeys the triangle inequality, we have $r' = d_F(K, Q) \leq d_F(P, Q) + d_F(K, P) = r^* + \delta r$.
Assume for the sake of contradiction that $r^* \leq (1 - 2\varepsilon')r$. Then, we have

$$r' \leq r^* + \delta r \leq (\delta + 1 - 2\varepsilon')\, r < r < 2r.$$

Therefore, $d(q_i, k_i^*) \leq r' < 2r$. Thus, there exists $k_i^* \in K$ that is at most a distance $2r$ away from q_i and the edge that k_i^* resides on is also at most $2r$ away from q_i. Hence, W_i is non-empty, and there exists $u_i \in W_i$ such that u_i and k_i^* share the same chord (edge) in K and $d_K(u_i, k_i^*) \leq \delta r$. In particular, there exists $u_i, v_i \in W_i$ where k_i^* is on the subcurve $K\langle u_i, v_i\rangle$, so that $d_K(u_i, k_i^*) \leq \delta r$, $d_K(v_i, k_i^*) \leq \delta r$, and $d_K(u_i, v_i) \leq \delta r$. See Figure 4.

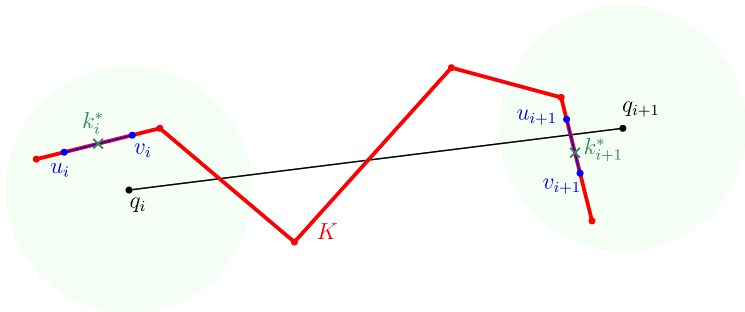

Figure 4 The point k_i^* marked with a green cross, and its immediate neighbour points u_i and v_i, marked with blue dots. Note that k_i^* is on the subcurve $K\langle u_i, v_i\rangle$, so that $d_K(u_i, k_i^*) \leq \delta r$, $d_K(v_i, k_i^*) \leq \delta r$, and $d_K(u_i, v_i) \leq \delta r$.

Consider $r_i' = d_F(K\langle u_i, u_{i+1}\rangle, q_i q_{i+1})$ when $q_i \in Q$ is matched to $u_i \in W_i \subset K$ and $q_{i+1} \in Q$ is matched to $u_{i+1} \in W_{i+1} \subset K$. Then

$$\begin{aligned}
r_i' &\leq d_F(K\langle k_i^*, k_{i+1}^*\rangle, q_i q_{i+1}) + d_F(K\langle k_i^*, k_{i+1}^*\rangle, K\langle u_i, u_{i+1}\rangle) \\
&\leq r' + d_F(K\langle k_i^*, k_{i+1}^*\rangle, K\langle u_i, u_{i+1}\rangle) \\
&\leq r' + d_F(k_i^* \circ K\langle k_i^*, u_{i+1}\rangle \circ u_{i+1}k_{i+1}^*, u_i k_i^* \circ K\langle k_i^*, u_{i+1}\rangle \circ u_{i+1}) \\
&\leq r' + \max\left\{d_F(k_i^*, u_i k_i^*), d_F(K\langle k_i^*, u_{i+1}\rangle, K\langle k_i^*, u_{i+1}\rangle), d_F(u_{i+1}k_{i+1}^*, u_{i+1})\right\} \\
&\leq r' + \max\left\{d_K(k_i^*, u_i), 0, d_K(k_{i+1}^*, u_{i+1})\right\} \\
&\leq r' + \max\left\{\delta r, 0, \delta r\right\} \\
&\leq r' + \delta r
\end{aligned}$$

where $\circ$ denotes the concatenation of polygonal curves. Therefore,

$$r_i' \leq r' + \delta r \leq (\delta + 1 - 2\varepsilon' + \delta)\, r = (2\delta + 1 - 2\varepsilon')\, r \leq (1 - \varepsilon')\, r.$$

Let C_i be the $(1+\delta)$-approximation of $d_F(K\langle u_i, u_{i+1}\rangle, q_i q_{i+1})$ returned by the data structure in Fact 6. Then, $r'_i \le C_i \le (1+\delta)r'_i$. Therefore, $C_i \le (1+\delta)r'_i < (1+\varepsilon')r'_i \le (1+\varepsilon')(1-\varepsilon')r < r$ for all $1 \le i \le m$. Hence, there is a directed edge from u_i to u_{i+1} in the layered directed graph, for all $1 \le i \le m-1$. In particular, $u_1 \dots u_m$ is a directed path from p_1 to p_n, which is a contradiction. We conclude that our assumption $r^* \le (1-2\varepsilon')r$ cannot hold, and it is correct to return $r^* > (1-2\varepsilon')r$ in the case where there is no directed path from p_1 to p_n.

We obtain the following theorem.

▶ **Theorem 7** (Fuzzy decider). *Given a positive real number r, $0 < \varepsilon < \frac{1}{2}$, and a c-packed curve P with n vertices in $\mathbb{R}^d$, one can construct a data structure in $O(n \log^{d+1} n + c\varepsilon^{-1} n \log n + \varepsilon^{-2d} \log^2(1/\varepsilon) n \log^2 n)$ time that uses $O(n \log^d n + c\varepsilon^{-1} n + \varepsilon^{-2d} \log^2(1/\varepsilon) n)$ space, so that given a query curve Q with m vertices, the data structure returns in $O(\log^d n + mc^2\varepsilon^{-6} \log n \log\log n)$ query time either (i) $d_F(P,Q) \le (1+\varepsilon'/2)r$ or (ii) $d_F(P,Q) > (1-2\varepsilon')r$.*

Proof. First, we summarise the preprocessing procedure. Let $\delta = \varepsilon/60$. We use Fact 2 to construct the simplification $K = \mathrm{simpl}(P, \delta r)$. We use Lemma 5 to construct a range searching data structure on K, and we use Fact 6 to construct an approximate distance data structure on K. Next, we summarise the query procedure. We query Lemma 5 to construct W_i for $1 \le i \le m$, we query Fact 6 to construct the edges between W_i and W_{i+1} for $1 \le i \le m-1$, and finally we run a breadth first search. We argued correctness in Section 2.2.3. It remains to analyse preprocessing time, space, and query time.

The preprocessing time of Fact 2, Lemma 5 and Fact 6 is $O(n \log^{d+1} n + c\delta^{-1} n \log n + \delta^{-2d} \log^2(1/\delta) n \log^2 n)$. The space of the data structures in Lemma 5 and Fact 6 is $O(n \log^d n + c\delta^{-1} n + \delta^{-2d} \log^2(1/\delta) n)$. Substituting $\delta^{-1} = O(\varepsilon^{-1})$ yields the stated preprocessing time and space.

We analyse the query time. Constructing the set W_i for all $1 \le i \le m$ takes $O(m(\log^d n + c\delta^{-2}))$ time, since using Lemma 5 to query the set of edges close to q_i takes $O(\log^d n + c\delta^{-1})$ time, and constructing evenly spaced points takes $O(c\delta^{-2})$ time. Since $|W_i| = O(c\delta^{-2})$, the number of pairs $\cup_{i=1}^{m-1}(W_i \times W_{i+1})$ is $O(mc^2\delta^{-4})$. Querying Fact 6 to decide whether there is a directed edge takes $O(\delta^{-2} \log n \log\log n)$ time per pair. In total, constructing the edges in the layered directed graph takes $O(mc^2\delta^{-6} \log n \log\log n)$ time. Running breadth first search takes $O(mc^2\delta^{-4})$ time. The total query time is $O(\log^d n + c^2\delta^{-6} m \log n \log\log n)$. Substituting $\delta^{-1} = O(\varepsilon^{-1})$ yields the stated query time. ◀

2.3 Complete approximate decider

The third step in the decision algorithm is to use the fuzzy decider to construct a complete approximate decider. Recall that, given ε, P, Q and r, the complete approximate decider returns either (*i*) $d_F(P,Q) \le r$, (*ii*) $d_F(P,Q) > r$, or (*iii*) a $(1+\varepsilon)$-approximation for $d_F(P,Q)$.

▶ **Theorem 8** (Complete approximate decider). *Given a positive real number r, $0 < \varepsilon < \frac{1}{2}$, and a c-packed curve P with n vertices in $\mathbb{R}^d$, one can construct a data structure in $O(n \log^{d+1} n + c\varepsilon^{-1} n \log n + \varepsilon^{-2d} \log^2(1/\varepsilon) n \log^2 n)$ time that uses $O(n \log^d n + c\varepsilon^{-1} n + \varepsilon^{-2d} \log^2(1/\varepsilon) n)$ space, so that given a query curve Q with m vertices in $\mathbb{R}^d$, the data structure returns in $O(\log^d n + mc^2\varepsilon^{-6} \log n \log\log n)$ query time either (i) $d_F(P,Q) \le r$, (ii) $d_F(P,Q) > r$, or (iii) a $(1+\varepsilon)$-approximation for $d_F(P,Q)$.*

Proof. Let $r_1 = \frac{1}{1+\varepsilon'/2} r$ and $r_2 = \frac{1}{1-2\varepsilon'} r$, where $\varepsilon' = \varepsilon/30$. First, given r_1, ε, and P, construct the data structure in Theorem 7, and query the data structure on Q. Second, given r_2, ε, and P, construct the data structure in Theorem 7, and query the data structure on Q. If the first query returns (i), we return (i). If both the first and second queries return (ii), we return (ii). Otherwise, if the first query returns (ii) and the second query returns (i), we return (iii). We prove correctness in three cases.

- The first query returns (i). Then by Theorem 7, $d_F(P,Q) \le (1+\varepsilon'/2)r_1 = r$, so returning (i) in the complete approximate decider is correct.
- Both the first and second queries return (ii). Then by Theorem 7, $d_F(P,Q) > (1-2\varepsilon')r_2 = r$, so returning (ii) in the complete approximate decider is correct.
- The first query returns (ii) and the second query returns (i). The first query implies $d_F(P,Q) > (1-2\varepsilon') \cdot r_1 = (1-2\varepsilon') \cdot \frac{1}{1+\varepsilon'/2} \cdot r$. The second query implies $d_F(P,Q) \le (1+\varepsilon'/2) \cdot r_1 = (1+\varepsilon'/2) \cdot \frac{1}{1-2\varepsilon'} \cdot r$. Putting these together, we have

  $$d_F(P,Q) \in \left(\frac{1-2\varepsilon'}{1+\varepsilon'/2} r, \frac{1+\varepsilon'/2}{1-2\varepsilon'} r \right].$$

  Note that

  $$\frac{\frac{1+\varepsilon'/2}{1-2\varepsilon'} r}{\frac{1-2\varepsilon'}{1+\varepsilon'/2} r} = \left(\frac{1+\varepsilon'/2}{1-2\varepsilon'} \right)^2 < ((1+\varepsilon'/2)(1+4\varepsilon'))^2 < (1+6\varepsilon')^2 < 1+30\varepsilon' = 1+\varepsilon.$$

  Hence, $\frac{1-2\varepsilon'}{1+\varepsilon'/2} r$ is a $(1+\varepsilon)$-approximation of $d_F(P,Q)$, so returning (iii) in the complete approximate decider is correct.

Finally, the preprocessing time, space, and query time follow from Theorem 7. ◀

This completes the decision version of the approximate Fréchet distance problem. Next, we consider the optimisation version of the approximate Fréchet distance problem.

3 Optimisation algorithm

In Section 3.1, we apply a binary search to compute the optimal simplification. In Section 3.2, we apply parametric search to compute the Fréchet distance. In both steps, we use the complete approximate decider in Theorem 8, which incurs an approximation factor of $(1+\varepsilon)$.

3.1 Approximating the optimal simplification

First, we provide an algorithm to compute the optimal simplification of P. In particular, the optimal simplification is $K^* = \text{simpl}(P, \delta r^*)$, where $\delta = \varepsilon/60$ and $r^* = d_F(P,Q)$. Our approach is to search over the critical values of the μ-simplification algorithm in Fact 2. A critical value of the μ-simplification algorithm is a value of μ where the simplification changes. Define the set of pairwise distances of P to be $L(P) = \{d(p_i, p_j) : 1 \le i < j \le n\}$. We can observe that the set of pairwise distances L breaks up the positive real line into $\binom{n}{2} + 1$ intervals, such that within each interval the μ-simplification does not change. This observation follows from the algorithm in Fact 2, and the same observation is made in Section 3.3.3 in [12]. Unfortunately, $|L| = O(n^2)$. To overcome this, we use approximate distance selection.

▶ **Fact 9** (Lemma 3.9 in [12]). *Given a set P of n points in $\mathbb{R}^d$, one can compute in $O(n \log n)$ time a set Z of $O(n)$ numbers, such that for any $y \in L(P)$, there exists numbers $x, x' \in Z$ such that $x \le y \le x' \le 2x$.*

We can refine Fact 9 to obtain Corollary 10. We replace the 8-WSPD in Lemma 3.9 of [12] with an $8/\varepsilon$-WSPD.

▶ **Corollary 10.** *Given a set P of n points in $\mathbb{R}^d$, one can compute in $O(n/\varepsilon^d + n\log n)$ time a set Z of $O(n/\varepsilon^d)$ numbers, such that for any $y \in L(P)$, there exists numbers $x, x' \in Z$ such that $x \le y \le x' \le (1+\varepsilon)x$.*

Next, we perform binary search on the set Z in Corollary 10. In particular, for $x \in Z$, we decide whether $\delta r^* < x$ or $\delta r^* > x$ by running the complete approximate decider in Theorem 8 on $r = x/\delta$, $\delta = \varepsilon/60$, P and Q. After $O(\log n)$ applications of the complete approximate decider, we obtain $\delta r^* \in [x, x']$ for a consecutive pair of elements $x, x' \in Z$. We have two cases. In the first case, we compute the optimal simplification of P, that is, $K^* = \text{simpl}(P, \delta r^*)$. In the second case, we compute a $(1+\varepsilon)$-approximation of $r^* = d_F(P, Q)$.

- If $x' > (1+\varepsilon)x$. By the contrapositive of Corollary 10, there is no $y \in L(P) \cap [x, x']$. In other words, within the interval $[x, x']$ the simplification of P does not change. Therefore, $K^* = \text{simpl}(P, x) = \text{simpl}(P, \delta r^*)$.
- If $x' \le (1+\varepsilon)x$. Therefore, x'/δ is a $(1+\varepsilon)$-approximation of $r^* = d_F(P, Q)$, as required.

Therefore, we can compute $K^* = \text{simpl}(P, \delta r^*)$, as otherwise we would have a $(1+\varepsilon)$-approximation of r^*. The running time is dominated by the $O(\log n)$ applications of the complete approximate decider.

3.2 Approximating the Fréchet distance

From Section 3.1, we computed the simplification $K^* = \text{simpl}(P, \delta r^*)$. Let $r_1^* = \frac{1}{1+\varepsilon'/2} r^*$, $r_2^* = \frac{1}{1-2\varepsilon'} r^*$. We can use the same procedure to compute the simplifications $K_1^* = \text{simpl}(P, \delta r_1^*)$ and $K_2^* = \text{simpl}(P, \delta r_2^*)$. If $K_1^* \ne K^*$, then there must be an element $x \in Z$ in the interval $[\delta r_1^*, \delta r^*]$, so x/δ would be a $(1+\varepsilon)$-approxmation of r^*. Therefore, $K^* = K_1^*$, and similarly, $K^* = K_2^*$.

We proceed with parametric search. Note that in Section 3.1, we did not apply parametric search to compute K^* due to efficiency reasons. It is not straightforward to parallelise Fact 2, moreover, since the simplification $K^* = K_1^* = K_2^*$ does not change during the execution of the parametric search, we can avoid reconstructing the data structures in Lemma 5 and Fact 6. We obtain the following theorem.

▶ **Theorem 11.** *Given $\varepsilon > 0$, a c-packed curve P in $\mathbb{R}^d$, and a general curve Q in $\mathbb{R}^d$, one can compute a $(1+\varepsilon)$-approximation of $d_F(P, Q)$ in $O(T_s T_p \log m)$ time, where*

$$T_s = n\log^{d+1} n + c\varepsilon^{-1} n\log n + \varepsilon^{-2d}\log^2(1/\varepsilon) n\log^2 n + mc^2\varepsilon^{-6}\log n\log\log n,$$
$$T_p = \log^d n + c\varepsilon^{-1} + \varepsilon^{-2}\log n\log\log n.$$

Proof. First, we summarise the preprocessing procedure. We compute the simplification $K^* = \text{simpl}(P, \delta r^*)$ using the procedure described in Section 3.1. We build the data structures in Lemma 5 and Fact 6 on the simplified curve K^*.

Second, we summarise the query procedure. Here, we use parametric search. We use the algorithm in Theorem 8 as both the decision algorithm and the simulated algorithm. We describe the simulated algorithm. Let r be the search parameter. Let $r_1 = \frac{1}{1+\varepsilon'/2} r$ and $r_2 = \frac{1}{1-2\varepsilon'} r$. We simulate the complete approximate decider by simulating the fuzzy decider in Theorem 7 on r_1 and r_2. We divide the simulation of the fuzzy decider on r_1 into three steps. First, we compute W_i by querying the data structure in Lemma 5. We use parametric search and the decision algorithm (Theorem 8) to resolve the critical values in the query. Second, we compute the directed edges from W_i to W_{i+1} by querying the data structure in Fact 6. We apply parametric search in the same way. Third, we run a breadth first search on the layered directed graph. There are no critical values in this step, so we do not need to apply parametric search. We repeat the simulation of the fuzzy decider on r_2. Finally, by parametric search, we return the optimal value r^*.

Third, we argue correctness. If Theorem 8 returns (iii) at any point, we obtain a $(1+\varepsilon)$-approximation of r^*, and we are done. If Theorem 8 never returns (iii) at any point, we will show that the decision algorithm and the simulated algorithm are both correct. The decision algorithm is correct since we either return $r^* \le r$ or $r^* > r$. We show the preprocessing and query procedures of the simulated algorithm are correct. In particular, we will show that we correctly simulate the execution of Theorem 8 as though $r = r^*$. The preprocessing procedure is correct, since $K^* = K_1^* = K_2^*$, so our data structures are correct for r_1^* and r_2^*. The query procedure is correct, since we can use the correct decision algorithm to resolve all critical values, and simulate the correct execution path as though $r = r^*$. Moreover, Theorem 8 (without (iii)) acts discontinuously at $r = r^*$, so r^* is a critical value of the simulated algorithm. Therefore, parametric search is able to locate r^* and return it.

Fourth, we analyse the running time. The preprocessing time is dominated by $O(\log n)$ calls to Theorem 8. The query time is dominated by parametric search. The running time of parametric search is $O(P_pT_p + T_pT_s \log P_p)$, where T_s is the sequential running time of the decision algorithm, P_p is the number of processors used in the simulated algorithm, and T_p is the number of parallel steps used by the simulated algorithm. The sequential running time is $T_s = O(n \log^{d+1} n + c\varepsilon^{-1} n \log n + \varepsilon^{-2d} \log^2(1/\varepsilon) n \log^2 n + mc^2\varepsilon^{-6} \log n \log\log n)$ by Theorem 8. The simulated algorithm can be efficiently parallelised. In particular, the simulated algorithm computes W_i by querying Lemma 5, and computes the directed edges from W_i to W_{i+1} by querying Lemma 6; these can be queried in parallel for all $1 \le i \le m$. Given $P_p = m$ processors, we can perform all of these queries in in $T_p = O(\log^d n + c\delta^{-1} + \delta^{-2} \log n \log\log n)$ parallel steps. The overall running time is dominated by $O(T_sT_p \log m)$, which the stated running time. ◀

We can simplify the running time if ε is constant.

▶ **Corollary 12.** *Given a constant $\varepsilon > 0$, a c-packed curve P with n vertices in $\mathbb{R}^d$, and a general curve Q with m vertices in $\mathbb{R}^d$, one can $(1+\varepsilon)$-approximate $d_F(P,Q)$ in $O(c^3(n+m)\log^{2d+1}(n)\log m)$ time.*

4 Conclusion

In this paper, we provide an $O(c^3(n+m)\log^{2d+1}(n)\log m)$ time algorithm to $(1+\varepsilon)$-approximate the Fréchet distance between two curves in $\mathbb{R}^d$, in the case when only one curve is c-packed and ε is constant. The running time is nearly-linear if c and d are also constant. An open problem is whether the running time can be improved, in particular, whether the dependence on ε, c, d, $\log n$ or $\log m$ can be reduced. Another open problem is whether we can obtain results for related problems when only one of the two curves is c-packed. Yet another open problem is whether similar results can be obtained for other realistic input curves. In particular, can the Fréchet distance be $(1+\varepsilon)$-approximated in subquadratic time when only one of the curves is κ-bounded, or when only one of the curves is ϕ-low density?

References

1 Helmut Alt and Michael Godau. Computing the Fréchet distance between two polygonal curves. *International Journal of Computational Geometry & Applications*, 05:75–91, 1995. doi:10.1142/s0218195995000064.

2 Karl Bringmann. Why walking the dog takes time: Fréchet distance has no strongly subquadratic algorithms unless SETH fails. In *Proceedings of the IEEE 55th Annual Symposium on Foundations of Computer Science*. IEEE, 2014. doi:10.1109/focs.2014.76.

3 Karl Bringmann and Marvin Künnemann. Improved approximation for Fréchet distance on c-packed curves matching conditional lower bounds. *International Journal of Computational Geometry & Applications*, 27:85–119, 2017. doi:10.1142/s0218195917600056.

4 Frederik Brüning, Jacobus Conradi, and Anne Driemel. Faster approximate covering of subcurves under the Fréchet distance. In *Proceedings of the 30th Annual European Symposium on Algorithms, ESA 2022*, volume 244 of *LIPIcs*, pages 28:1–28:16. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2022. doi:10.4230/LIPICS.ESA.2022.28.

5 Kevin Buchin, Maike Buchin, Joachim Gudmundsson, Aleksandr Popov, and Sampson Wong. Map-matching queries under Fréchet distance on low-density spanners. In *Proceedings of the 40th International Symposium on Computational Geometry, SoCG 2024*, volume 293 of *LIPIcs*, pages 27:1–27:15. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2024. doi:10.4230/LIPICS.SOCG.2024.27.

6 Kevin Buchin, Maike Buchin, Wouter Meulemans, and Wolfgang Mulzer. Four Soviets walk the dog: Improved bounds for computing the Fréchet distance. *Discrete & Computational Geometry*, 58(1):180–216, 2017. doi:10.1007/s00454-017-9878-7.

7 Kevin Buchin, Tim Ophelders, and Bettina Speckmann. SETH says: Weak Fréchet distance is faster, but only if it is continuous and in one dimension. In *Proceedings of the 30th Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2019*, pages 2887–2901. SIAM, 2019. doi:10.1137/1.9781611975482.179.

8 Daniel Chen, Anne Driemel, Leonidas J. Guibas, Andy Nguyen, and Carola Wenk. Approximate map matching with respect to the Fréchet distance. In *Proceedings of the 13th Workshop on Algorithm Engineering and Experiments, ALENEX 2011*, pages 75–83. SIAM, 2011. doi:10.1137/1.9781611972917.8.

9 Jacobus Conradi, Anne Driemel, and Benedikt Kolbe. $(1+\epsilon)$-ANN data structure for curves via subspaces of bounded doubling dimension. *Comput. Geom. Topol.*, 3(2):6:1–6:22, 2024. URL: https://www.cgt-journal.org/index.php/cgt/article/view/45.

10 Jacobus Conradi, Anne Driemel, and Benedikt Kolbe. Revisiting the Fréchet distance between piecewise smooth curves. *CoRR*, abs/2401.03339, 2024. doi:10.48550/arXiv.2401.03339.

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

12 Anne Driemel, Sariel Har-Peled, and Carola Wenk. Approximating the Fréchet distance for realistic curves in near linear time. *Discrete & Computational Geometry*, 48(1):94–127, 2012. doi:10.1007/s00454-012-9402-z.

13 M. Maurice Fréchet. Sur quelques points du calcul fonctionnel. *Rendiconti del Circolo Matematico di Palermo*, 22(1):1–72, 1906. doi:10.1007/bf03018603.

14 Joachim Gudmundsson, Zijin Huang, André van Renssen, and Sampson Wong. Computing a subtrajectory cluster from c-packed trajectories. In *Proceedings of the 34th International Symposium on Algorithms and Computation, ISAAC 2023*, volume 283 of *LIPIcs*, pages 34:1–34:15. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2023. doi:10.4230/LIPICS.ISAAC.2023.34.

15 Joachim Gudmundsson, Michael Mai, and Sampson Wong. Approximating the Fréchet distance when only one curve is c-packed. *CoRR*, abs/2407.05114, 2024. doi:10.48550/arXiv.2407.05114.

16 Joachim Gudmundsson, Martin P. Seybold, and Sampson Wong. Map matching queries on realistic input graphs under the Fréchet distance. *ACM Trans. Algorithms*, 20(2):14, 2024. doi:10.1145/3643683.

17 Joachim Gudmundsson, Yuan Sha, and Sampson Wong. Approximating the packedness of polygonal curves. *Comput. Geom.*, 108:101920, 2023. doi:10.1016/J.COMGEO.2022.101920.

18 Joachim Gudmundsson and Michiel H. M. Smid. Fast algorithms for approximate Fréchet matching queries in geometric trees. *Comput. Geom.*, 48(6):479–494, 2015. doi:10.1016/J.COMGEO.2015.02.003.

19 Sariel Har-Peled and Benjamin Raichel. The Fréchet distance revisited and extended. *ACM Trans. Algorithms*, 10(1):3:1–3:22, 2014. doi:10.1145/2532646.

20 Sariel Har-Peled and Timothy Zhou. How packed is it, really? *CoRR*, abs/2105.10776, 2021. arXiv:2105.10776.

21 Minghui Jiang, Ying Xu, and Binhai Zhu. Protein structure–structure alignment with discrete Fréchet distance. *Journal of Bioinformatics and Computational Biology*, 06(01):51–64, 2008. doi:10.1142/s0219720008003278.

22 Richard J. Kenefic. Track clustering using Fréchet distance and minimum description length. *Journal of Aerospace Information Systems*, 11(8):512–524, 2014. doi:10.2514/1.i010170.

23 Patrick Laube. *Computational Movement Analysis*. Springer International Publishing, 2014. doi:10.1007/978-3-319-10268-9.

24 Peter Ranacher and Katerina Tzavella. How to compare movement? A review of physical movement similarity measures in geographic information science and beyond. *Cartography and Geographic Information Science*, 41(3):286–307, 2014. doi:10.1080/15230406.2014.890071.

25 Otfried Schwarzkopf and Jules Vleugels. Range searching in low-density environments. *Information Processing Letters*, 60(3):121–127, 1996. doi:10.1016/s0020-0190(96)00154-8.

26 Kevin Toohey and Matt Duckham. Trajectory similarity measures. *SIGSPATIAL Special*, 7(1):43–50, 2015. doi:10.1145/2782759.2782767.

27 Ivor van der Hoog, Eva Rotenberg, and Sampson Wong. Data structures for approximate discrete Fréchet distance. *CoRR*, abs/2212.07124, 2022. doi:10.48550/arXiv.2212.07124.

28 Haozhou Wang, Han Su, Kai Zheng, Shazia Sadiq, and Xiaofang Zhou. An effectiveness study on trajectory similarity measures. In *Proceedings of the 24th Australasian Database Conference – Volume 137*, ADC '13, pages 13–22. Australian Computer Society, Inc., 2013.

29 Tim Wylie and Binhai Zhu. Protein chain pair simplification under the discrete Fréchet distance. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 10(6):1372–1383, 2013. doi:10.1109/tcbb.2013.17.