The Discrete and Semi-continuous Fréchet Distance with Shortcuts via Approximate Distance Counting and Selection

Rinat Ben Avraham† Omrit Filtser‡ Haim Kaplan§ Matthew J. Katz¶ Micha Sharir∥

Abstract

The Fréchet distance is a well studied similarity measure between curves. The discrete Fréchet distance is an analogous similarity measure, defined for two sequences of \(m\) and \(n\) points, where the points are usually sampled from input curves. We consider a variant, called the discrete Fréchet distance with shortcuts, which captures the similarity between (sampled) curves in the presence of outliers. When shortcuts are allowed only in one noise-containing curve, we give a randomized algorithm that runs in \(O((m + n)^{6/5+\epsilon})\) expected time, for any \(\epsilon > 0\). When shortcuts are allowed in both curves, we give an \(O((m^{2/3}n^{2/3} + m + n)\log^3(m + n))\)-time deterministic algorithm.

We also consider the semi-continuous Fréchet distance with one-sided shortcuts, where we have a sequence of \(m\) points and a polygonal curve of \(n\) edges, and shortcuts are allowed only in the sequence. We show that this problem can be solved in randomized expected time \(O((m + n)^{2/3}m^{2/3}n^{1/3}\log(m + n))\).

Our techniques are novel and may find further applications. One of the main new technical results is: Given two sets of points \(A\) and \(B\) in the plane and an interval \(I\), we develop an algorithm that decides whether the number of pairs \((x, y)\) in \(A \times B\) whose distance \(\text{dist}(x, y)\) is in \(I\), is less than some given threshold \(L\). The running time of this algorithm decreases as \(L\) increases. In case there are more than \(L\) pairs of points whose distance is in \(I\), we can get a small sample of pairs that contains a pair at approximate median distance (i.e., we can approximately “bisect” \(I\)). We combine this procedure with additional ideas to search, with a small overhead, for the optimal one-sided Fréchet distance with shortcuts, using a very fast decision procedure. We also show how to apply this technique for approximating distance selection (with respect to rank), and a somewhat more involved variant of this technique is used in the solution of the semicontinuous Fréchet distance with one-sided shortcuts. In general, the new technique can apply to optimization problems for which the decision procedure is very fast but standard techniques like parametric search makes the optimization algorithm substantially slower.

---

*Work by Omrit Filtser has been partially supported by the Lynn and William Frankel Center for Computer Sciences. Work by Haim Kaplan has been supported by Israel Science Foundation grant no. 822-10, and German-Israeli Foundation for Scientific Research and Development (GIF) grant no. 1161/2011, and Israeli Centers of Research Excellence (I-CORE) program, (Center No. 4/11). Work by M. Katz has been partially supported by grant 1045/10 from the Israel Science Foundation, and grant 2010074 from the United States – Israel Binational Science Foundation. Work by Micha Sharir has been supported by Grant 892/13 from the Israel Science Foundation, by the Israeli Centers of Research Excellence (I-CORE) program (Center No. 4/11), and by the Hermann Minkowski-MINERVA Center for Geometry at Tel Aviv University. Work by Micha Sharir and Rinat Ben Avraham has been supported by Grant 2012/229 from the U.S.-Israeli Binaional Science Foundation. A preliminary version of this paper has been accepted to SOCG’14 [3].

†School of Computer Science, Tel Aviv University, Tel Aviv 69978, Israel; rinatba@gmail.com
‡Department of Computer Science, Ben-Gurion University, Beer-Sheva 84105, Israel; omritna@post.bgu.ac.il
§School of Computer Science, Tel Aviv University, Tel Aviv 69978, Israel; haimk@post.tau.ac.il
¶Department of Computer Science, Ben-Gurion University, Beer-Sheva 84105, Israel; matya@cs.bgu.ac.il
∥School of Computer Science, Tel Aviv University, Tel Aviv 69978, Israel; michas@post.tau.ac.il
1 Introduction

Consider a person and a dog connected by a leash, each walking along a curve from its starting point to its end point. Both are allowed to control their speed but they cannot backtrack. The Fréchet distance between the two curves is the minimum length of a leash that is sufficient for traversing both curves in this manner. The discrete Fréchet distance replaces the curves by two sequences of points $A = (a_1, \ldots, a_m)$ and $B = (b_1, \ldots, b_n)$, and replaces the person and the dog by two frogs, the $A$-frog and the $B$-frog, initially placed at $a_1$ and $b_1$, respectively. At each move, the $A$-frog or the $B$-frog (or both) jumps from its current point to the next. The frogs are not allowed to backtrack. We are interested in the minimum length of a “leash” that connects the frogs and allows the $A$-frog and the $B$-frog to get to $a_m$ and $b_n$, respectively. More formally, for a given length $\delta$ of the leash, a jump is allowed only if the distances between the two frogs before and after the jump are both at most $\delta$; the discrete Fréchet distance between $A$ and $B$, denoted by $\delta^*_{F}(A,B)$, is then the smallest $\delta > 0$ for which there exists a sequence of jumps that brings the frogs to $a_m$ and $b_n$, respectively.

The Fréchet distance and the discrete Fréchet distance are used as similarity measures between curves and sampled curves, respectively, in many applications. Among these are speech recognition [21], signature verification [24], matching of time series in databases [20], map-matching of vehicle tracking data [6, 14, 26], and analysis of moving objects [7, 8].

In many of these applications the curves or the sampled sequences of points are generated by physical sensors, such as GPS. These sensors may generate inaccurate measurements, which we refer to as outliers. The Fréchet distance and the discrete Fréchet distance are bottleneck (min-max) measures, and are therefore sensitive to outliers, and may fail to capture the similarity between the curves when there are outliers, because the large distance from an outlier to the other curve might determine the Fréchet distance, making it much larger than the distance without the outliers.

In order to handle outliers, Driemel and Har-Peled [15] introduced the (continuous) Fréchet distance with shortcuts. They considered piecewise linear curves and allowed (only) the dog to take shortcuts by walking from a vertex $v$ to any succeeding vertex $w$ along the straight segment connecting $v$ and $w$. This “one-sided” variant allows one to “ignore” subcurves of one (noisy) curve that substantially deviate from the other (more reliable) curve. Driemel and Har-Peled gave efficient approximation algorithms for the Fréchet distance in such scenarios; these are reviewed in more detail later on.

Driven by the same motivation of reducing sensitivity to outliers, we define two variants of the discrete Fréchet distance with shortcuts. In the one-sided variant, we allow the $A$-frog to jump to any point that comes later in its sequence, rather than just to the next point. The $B$ frog has to visit all the $B$ points in order, as in the standard discrete Fréchet distance problem. However, we add the restriction that only a single frog is allowed to jump in each move (see below for more details). As in the standard discrete Fréchet distance, for a leash of length $\delta$ such a jump is allowed only if the distances between the two frogs before and after the jump are both at most $\delta$. The one-sided discrete Fréchet distance with shortcuts, denoted as $\delta^+_{F}(A,B)$, is the smallest $\delta > 0$ for which there exists such a sequence of jumps that brings the frogs to $a_m$ and $b_n$, respectively.

We also define the two-sided discrete Fréchet distance with shortcuts, denoted as $\delta^\pm_{F}(A,B)$, to be the smallest $\delta > 0$ for which there exists a sequence of jumps, where both frogs are allowed to skip points as long as the distances between the two frogs before and after the jump are both at most $\delta$. Here too, we allow only one of the frogs to jump at each move.

In the (standard) discrete Fréchet distance, the frogs can make simultaneous jumps, each to its next point. Here though simultaneous jumps make the problem degenerate as it is possible for the frogs to jump from
Our results. In this paper we give efficient algorithms for computing the discrete Fréchet distance with one-sided and two-sided shortcuts. The structure of the one-sided problem allows us to decide whether the distance is no larger than a given δ, in \(O(n + m)\) time, and the challenge is to search for the optimum, using this fast decision procedure, with a small overhead. The naive approach would be to use the \(O((m^{2/3}n^{2/3} + m + n) \log(m + n))\)-time distance selection procedure of [19], which would make the running time \(\Omega((m^{2/3}n^{2/3} + m + n) \log(m + n))\), much higher than the linear cost of the decision procedure.

To tighten this gap, we develop an algorithm that, given an interval \([\alpha, \beta]\) and a parameter \(L\), decides, with high probability and in \(O((m + n)^{4/3+\epsilon}/L^{1/3} + m + n)\) time, whether the number of pairs in \(A \times B\) of distance in \([\alpha, \beta]\) is at most \(L\). Furthermore, if this number is larger than \(L\), our algorithm provides a sample of these pairs, of logarithmic size, that contains, with high probability, a pair at approximate median distance (in the middle three quarters of the distances in \([\alpha, \beta]\)). We combine this algorithm with a binary search to obtain a procedure that produces an interval that contains the optimal distance as well as at most \(L\) other distances. In addition, we give a technique to use the decision procedure in order to find the optimal value among these \(L\) remaining distances in \(O((m + n)L^{1/2} \log(m + n))\) time. As \(L\) increases, the first stage becomes faster and the second stage becomes slower. Choosing \(L\) to balance the two gives us an algorithm for the one-sided Fréchet distance with shortcuts that runs in \(O((m + n)^{6/5+\epsilon})\) time for any \(\epsilon > 0\).

We believe that this technique is of independent interest, beyond the scope of computing the one-sided Fréchet distance with shortcuts, and that it may be applicable to other optimization problems over pairwise distances. We give two such additional applications. The first application, given in Corollary 4.4, is rank-based approximation of the \(k\)th smallest distance. That is, given \(k\) and \(L \leq k\), we give an algorithm for finding a distance which is the \(\kappa\)th smallest distance, for some \(k - L \leq \kappa \leq k + L\), that runs in \(O((m + n)^{4/3+\epsilon}/L^{1/3} + m + n)\) time.

Our second application is a semi-continuous version of the one-sided Fréchet distance with shortcuts. In this problem \(A\) is a sequence of \(m\) points and \(f \subseteq \mathbb{R}^2\) is a polygonal curve of \(n\) edges. A frog has to jump over the points in \(A\), connected by a leash to a person who walks on \(f\). The frog can make shortcuts and skip points, but the person must traverse \(f\) continuously. The frog and the person cannot backtrack. We want to compute the minimum length of a leash that allows the frog and the person to get to their final positions in such a scenario. In Section 6 we present an algorithm that runs in \(O((m + n)^{2/3}m^{2/3}n^{1/3} \log(m + n))\) expected time for this problem. While less efficient than the fully discrete version, it is still significantly subquadratic.

For the two-sided version we take a different approach. More specifically, we implement the decision procedure by using an implicit compact representation of all pairs in \(A \times B\) at distance at most \(\delta\) as the disjoint union of complete bipartite cliques [19]. This representation allows us to maintain the pairs reachable by the frogs with a leash of length at most \(\delta\) implicitly and efficiently. The cost of the decision procedure is \(O((m^{2/3}n^{2/3} + m + n) \log^2(m + n))\), which is comparable with the cost of the distance selection procedure of [19], as mentioned above. We can then run a binary search for the optimal distance, using this distance selection procedure. The resulting algorithm runs in \(O((m^{2/3}n^{2/3} + m + n) \log^3(m + n))\) time and requires \(O((m^{2/3}n^{2/3} + m + n) \log(m + n))\) space.

Interestingly, the algorithms developed for these variants of the discrete Fréchet distance problem are sublinear in the size of \(A \times B\) and way below the slightly subquadratic bound for the discrete Fréchet distance, recently obtained in [1].
In principle, the algorithm for the one-sided Fréchet distance with shortcuts can be generalized to work in higher dimensions. See a remark in Section 4.

**Background.** The Fréchet distance and its variants have been extensively studied in the past two decades. Alt and Godau [2] showed that the Fréchet distance of two planar polygonal curves with a total of \( n \) edges can be computed, using dynamic programming, in \( O(n^2 \log n) \) time. Eiter and Mannila [16] showed that the discrete Fréchet distance in the plane can be computed, also using dynamic programming, in \( O(mn) \) time. Buchin et al. [9] recently improved the bound of Alt and Godau and showed how to compute the Fréchet distance in \( O(n^2 (\log n)^{1/2} (\log \log n)^{3/2}) \) time on a pointer machine, and in \( O(n^2 (\log \log n)^2) \) time on a word RAM [9]. Agarwal et al. [1] showed how to compute the discrete Fréchet distance in \( O(mn \log \log n \log n) \) time.

As already noted, the (one-sided) continuous Fréchet distance with shortcuts was first studied by Driemel and Har-Peled [15]. They considered the problem where shortcuts are allowed only between vertices of the noise-containing curve, in the manner outlined above, and gave approximation algorithms for solving two variants of this problem. In the first variant, any number of shortcuts is allowed, and in the second variant, the number of allowed shortcuts is at most \( k \), for some \( k \in \mathbb{N} \). Their algorithms work efficiently only for \( c \)-packed polygonal curves; these are curves that behave “nicely” and are assumed to be the input in practice. Both algorithms compute a \((3 + \varepsilon)\)-approximation of the Fréchet distance with shortcuts between two \( c \)-packed polygonal curves and both run in near-linear time (ignoring the dependence on \( \varepsilon \)). Buchin et al. [11] consider a more general version of the (one-sided) continuous Fréchet distance with shortcuts, where shortcuts are allowed between any pair of points of the noise-containing curve. They show that this problem is NP-Hard. They also give a 3-approximation algorithm for the decision version of this problem that runs in \( O(n^3 \log n) \) time.

In contrast with the results just reviewed, our results are somewhat surprising, as they demonstrate that both variants of the discrete Fréchet distance with shortcuts are easier to compute (exactly, with no restriction on the input) than all previously studied variants of the Fréchet distance.

We also note that there have been several other works that treat outliers in different ways. One such result is of Buchin et al. [10], who considered the partial Fréchet similarity problem, where one is given two curves \( f \) and \( g \), and a distance threshold \( \delta \), and the goal is to maximize the total length of the portions of \( f \) and \( g \) that are matched (using the Fréchet distance paradigm) with \( L_p \)-distance smaller than \( \delta \). They gave an algorithm that solves this problem in \( O(mn(m + n) \log (mn)) \) time, under the \( L_1 \) or \( L_\infty \) norm. The definition of the partial Fréchet similarity aims at situations where the extent of a prerequired similarity is known (and given by the distance threshold \( \delta \)), and we wish to know how much (and which parts) of the curves are similar to this extent. The definition of the (one-sided and two-sided) Fréchet with shortcuts is practically used in cases where we have a pre-assumption that the curves are similar, up to the existence of (not too many) outliers, and we want to estimate the magnitude of this similarity, eliminating the outliers. Since we assume that the points are sampled along curves that we want to match, our algorithms are applicable to any scenario in which the continuous Fréchet with shortcuts is applicable. Practical implementations of Fréchet distance algorithms that are made for experiments on real data in map matching applications, remove outliers from the data set [14, 26]. In another map matching application, Brakatsoulas et al. [6] define the notion of integral Fréchet distance to deal with outliers. This distance measure averages over certain distances instead of taking the maximum. Bereg et al. [4] and then Wylie and Zhao [27] considered the discrete Fréchet distance in biological context, for protein (backbone) structure alignment and comparison. They use pair simplification of the protein backbones, that can be interpreted as making shortcuts while comparing them under the discrete Fréchet distance.
2 Preliminaries

We now give a formal definition of the discrete Fréchet distance and its variants.

Let \( A = (a_1, \ldots, a_m) \) and \( B = (b_1, \ldots, b_n) \) be two sequences of \( m \) and \( n \) points, respectively, in the plane. Let \( G(V, E) \) denote a graph whose vertex set is \( V \) and edge set is \( E \), and let \( \| \cdot \| \) denote the Euclidean norm. Fix a distance \( \delta > 0 \), and define the following three directed graphs \( G_\delta = G(A \times B, E_\delta) \), \( G^-_\delta = G(A \times B, E^-_\delta) \), and \( G^+_\delta = G(A \times B, E^+_\delta) \), where

\[
E_\delta = \left\{ \left( (a_i, b_j), (a_{i+1}, b_{j+1}) \right) \mid \|a_i - b_j\|, \|a_{i+1} - b_{j+1}\| \leq \delta \right\} \cup \left\{ \left( (a_i, b_j), (a_i, b_{j+1}) \right) \mid \|a_i - b_j\|, \|a_i - b_{j+1}\| \leq \delta \right\},
\]

\[
E^-_\delta = \left\{ \left( (a_i, b_j), (a_k, b_j) \right) \mid k > i, \|a_i - b_j\|, \|a_k - b_j\| \leq \delta \right\} \cup \left\{ \left( (a_i, b_j), (a_i, b_{j+1}) \right) \mid \|a_i - b_j\|, \|a_i - b_{j+1}\| \leq \delta \right\},
\]

\[
E^+_\delta = \left\{ \left( (a_i, b_j), (a_k, b_j) \right) \mid k > i, \|a_i - b_j\|, \|a_k - b_j\| \leq \delta \right\} \cup \left\{ \left( (a_i, b_j), (a_i, b_l) \right) \mid l > j, \|a_i - b_j\|, \|a_i - b_l\| \leq \delta \right\}.
\]

For each of these graphs we say that a position \((a_i, b_j)\) is a reachable position if \((a_i, b_j)\) is reachable from \((a_1, b_1)\) in the respective graph. Then the discrete Fréchet distance (DFD for short) \( \delta^*_F(A, B) \) is the smallest \( \delta > 0 \) for which \((a_m, b_n)\) is a reachable position in \( G_\delta \). Similarly, the one-sided Fréchet distance with shortcuts (one-sided DFDS for short) \( \delta^-_F(A, B) \) is the smallest \( \delta > 0 \) for which \((a_m, b_n)\) is a reachable position in \( G^-_\delta \), and the two-sided Fréchet distance with shortcuts (two-sided DFDS for short) \( \delta^+_F(A, B) \) is the smallest \( \delta > 0 \) for which \((a_m, b_n)\) is a reachable position in \( G^+_\delta \).

3 Decision procedure for the one-sided DFDS

We first consider the corresponding decision problem. That is, given a value \( \delta > 0 \), we wish to decide whether \( \delta^*_F(A, B) \leq \delta \) (we ignore the issue of discrimination between the cases of strict inequality and equality, in the decision procedures of both the one-sided variant and the two-sided variant, since this will be handled in the optimization procedures, described later).

Let \( M \) be the matrix whose rows correspond to the elements of \( A \) and whose columns correspond to the elements of \( B \) and \( M_{i,j} = 1 \) if \( \|a_i - b_j\| \leq \delta \), and \( M_{i,j} = 0 \) otherwise. Consider first the DFD variant (no shortcuts allowed), in which, at each move, exactly one of the frogs has to jump to the next point. Suppose that \((a_i, b_j)\) is a reachable position of the frogs. Then, necessarily, \( M_{i,j} = 1 \). If \( M_{i+1,j} = 1 \) then the next move can be an upward move in which the \( A \)-frog moves from \( a_i \) to \( a_{i+1} \), and if \( M_{i,j+1} = 1 \) then the next move can be a right move in which the \( B \)-frog moves from \( b_j \) to \( b_{j+1} \). It follows that to determine whether \( \delta^*_F(A, B) \leq \delta \), we need to determine whether there is a right-upward staircase of ones in \( M \) that starts at \( M_{1,1} \), ends at \( M_{m,n} \), and consists of a sequence of interweaving upward moves and right moves (see Figure 1(a)).

In the one-sided version of DFDS, given a reachable position \((a_i, b_j)\) of the frogs, the \( A \)-frog can move to any point \( a_k, k > i \), for which \( M_{k,j} = 1 \); this is a skipping upward move in \( M \) which starts at \( M_{i,j} = 1 \), skips over \( M_{i+1,j}, \ldots, M_{k-1,j} \) (some of which may be 0), and reaches \( M_{k,j} = 1 \). However, in this
If there exists a semi-sparse staircase that ends at $M_{i,j+1} = 1$ (otherwise no move of the $B$-frog is possible at this position). Determining whether $\delta_F^-(A, B) \leq \delta$ corresponds to deciding whether there is a semi-sparse staircase of ones in $M$ that starts at $M_{1,1}$, ends at $M_{m,n}$, and consists of an interweaving sequence of skipping upward moves and (consecutive) right moves (see Figure 1b).

Assume that $M_{1,1} = 1$ and $M_{m,n} = 1$; otherwise, we can immediately conclude that $\delta_F^-(A, B) > \delta$ and terminate the decision procedure. From now on, whenever we refer to a semi-sparse staircase, we mean a semi-sparse staircase of ones in $M$ starting at $M_{1,1}$, as defined above, but without the requirement that it ends at $M_{m,n}$.

- $S \leftarrow \langle M_{1,1} \rangle$
- $i \leftarrow 1$, $j \leftarrow 1$
- While $(i < m$ or $j < n)$ do
  - If (a right move is possible) then
    - Make a right move and add position $M_{i,j+1}$ to $S$
    - $j \leftarrow j + 1$
  - Else
    - If (a skipping upward move is possible) then
      - Move upwards to the first (i.e., lowest) position $M_{k,j}$, with $k > i$ and $M_{k,j} = 1$, and add $M_{k,j}$ to $S$
      - $i \leftarrow k$
    - Else
      - Return $\delta_F^-(A, B) \leq \delta$
- Return $\delta_F^-(A, B) > \delta$

Figure 2: Decision procedure for the one-sided discrete Fréchet distance with shortcuts.

The algorithm of Figure 2 that implements the decision procedure, constructs a semi-sparse staircase $S$ by always making a right move if possible. The correctness of the decision procedure is established by the following lemma.

**Lemma 3.1.** If there exists a semi-sparse staircase that ends at $M_{m,n}$, then $S$ also ends at $M_{m,n}$. Hence $S$ ends at $M_{m,n}$ if and only if $\delta_F^-(A, B) \leq \delta$.

**Proof.** Let $S'$ be a semi-sparse staircase that ends at $M_{m,n}$. We think of $S'$ as a sequence of possible positions (i.e., 1-entries) in $M$. Note that $S'$ has at least one position in each column of $M$, since skipping is not allowed when moving rightwards. We claim that for each position $M_{k,j}$ in $S'$, there exists a position
\( M_{i,j} \) in \( S \), such that \( i \leq k \). This, in particular, implies that \( S \) reaches the last column. If \( S \) reaches the last column, we can continue it and reach \( M_{m,n} \) by a sequence of skipping upward moves (or just by one such move), so the lemma follows.

We prove the claim by induction on \( j \). It clearly holds for \( j = 1 \) as both \( S \) and \( S' \) start at \( M_{1,1} \). We assume then that the claim holds for \( j = \ell - 1 \), and establish it for \( \ell \). That is, assume that if \( S' \) contains an entry \( M_{i,\ell-1} \), then \( S \) contains \( M_{i,\ell-1} \) for some \( i \leq k \). Let \( M_{k',\ell} \) be the lowest position of \( S' \) in column \( \ell \); clearly, \( k' \geq k \). We must have \( M_{k',\ell-1} = 1 \) (as the only way to move from a column to the next is by a right move). If \( M_{i,\ell} = 1 \) then \( M_{i,\ell} \) is added to \( S \) by making a right move, and \( i \leq k \leq k' \) as required. Otherwise, \( S \) is extended by a sequence of skipping upward moves in column \( \ell - 1 \) followed by a right move between \( M_{i',\ell-1} \) and \( M_{i',\ell} \) where \( i' \) is the smallest index \( \geq i \) for which both \( M_{i',\ell-1} \) and \( M_{i',\ell} \) are one. But since \( i \leq k' \) and \( M_{k',\ell-1} \) and \( M_{k',\ell} \) are both 1, we get that \( i' \leq k' \), as required. \qed

**Running time.** The entries of \( M \) that the decision procedure tests form a row- and column-monotone path, with an additional entry to the right for each upward turn of the path. (This also takes into account the 0-entries of \( M \) that are inspected during a skipping upward move.) Therefore it runs in \( O(m + n) \) time.

### 4 One-sided DFDS optimization via approximate distance counting and selection

We now show how to use the decision procedure of Figure 2 to solve the optimization problem of the one-sided discrete Fréchet distance with shortcuts. This is based on the algorithms provided in Lemma 4.1 and Lemma 4.2 given below. We believe that the algorithm of Lemma 4.1 is of independent interest, and give another application of it to a different distance-related problem in Corollary 4.4 below.

First note that if we increase \( \delta \) continuously, the set of 1-entries of \( M \) can only grow, and this can only happen when \( \delta \) is a distance between a point of \( A \) and a point of \( B \). Performing a binary search over the \( O(mn) \) pairwise distances of pairs in \( A \times B \) can be done using the distance selection algorithm of [19]. This will be the method of choice for the two-sided DFDS problem, treated in Section 3. Here however, this procedure, which takes \( O(m^{2/3}n^{2/3}\log^3(m + n)) \) time is rather excessive when compared to the linear cost of the decision procedure. While solving the optimization problem in close to linear time is still a challenging open problem, we manage to improve the running time considerably, to \( O((m + n)^{6/5+\varepsilon}) \), for any \( \varepsilon > 0 \).

We first present the two main subproblems that the algorithm uses as independent building blocks, in the following two lemmas.

**Lemma 4.1.** Given a set \( A \) of \( m \) points and a set \( B \) of \( n \) points in the plane, an interval \( \langle \alpha, \beta \rangle \subset \mathbb{R} \), and parameters \( 0 < L \leq mn \) and \( \varepsilon > 0 \), we can determine, with high probability, whether \( \langle \alpha, \beta \rangle \) contains at most \( L \) distances between pairs in \( A \times B \). If \( \langle \alpha, \beta \rangle \) contains more than \( L \) such distances, we return a sample of \( O(\log(m + n)) \) pairs, so that, with high probability, at least one of these pairs determines an approximate median (in the middle three quarters) of the pairwise distances that lie in \( \langle \alpha, \beta \rangle \). Our algorithm runs in \( O((m + n)^{4/3+\varepsilon}/L^{1/3} + m + n) \) time and uses \( O((m + n)^{4/3+\varepsilon}/L^{1/3} + m + n) \) space.

**Proof.** We partially construct a batched range counting data structure for representing (some of) the pairs \( (p,q) \in A \times B \) whose distance lies in \( \langle \alpha, \beta \rangle \), as the edge-disjoint union of complete bipartite graphs. If we build the complete data structure, it will require \( O((m + n)^{4/3+\varepsilon}) \) time and \( O((m + n)^{4/3+\varepsilon}) \) storage (similar to the parameters of the structure of [19] used in Section 5). Since this is too expensive, we run
the construction until it reaches a level where the size of each subproblem is at most $L$ (details will follow shortly), and then stop. The pairwise distances of $A \times B$ that fall in $(\alpha, \beta)$ are now of two types, those recorded in the complete bipartite graphs that we have constructed, and those where the two points belong to the same remaining “leaf” subproblem. We estimate the number of distances of the latter type using an appropriate random sample of points. As we show later, this also allows us to return a sample that contains an approximate median of the pairwise distances in $(\alpha, \beta)$, with high probability, if the number of these distances is indeed larger than $L$.

In more detail, we proceed as follows. Let $C$ denote the collection of the circles bounding the $(\alpha, \beta)$-annuli that are centered at the points of $A$. We choose a sufficiently large constant parameter $1 \leq r \leq m$, and construct a $(1/r)$-cutting for $C$. That is, for a suitable constant $c$, we partition the plane into $k \leq cr^2$ cells $\Delta_1, \ldots, \Delta_k$, each of constant description complexity, so that each $\Delta_i$ is crossed by at most $m/r$ boundaries of the annuli, and each $\Delta_i$ contains at most $n/r^2$ points of $B$. This can be done in $O((m+n)r)$ deterministic time, as in [12, 13, 23]. We then dualize the roles of $A$ and $B$, in each cell $\Delta_i$ separately, where the set $B_{\Delta_i}$ of the at most $n/r^2$ points of $B$ in $\Delta_i$ becomes a set of $(\alpha, \beta)$-annuli centered at these points, and the set $A_{\Delta_i}$ of the at most $m/r$ points of $A$ whose annuli boundaries cross $\Delta_i$ is now regarded as a set of points. We now construct, for each $\Delta_i$, a $(1/r)$-cutting in this dual setting. We obtain a total of at most $c^2r^4$ subproblems, each involving at most $m/r^3$ points of $A$ and at most $n/r^3$ points of $B$.

In both the primal and dual stages, we output a collection of complete bipartite graphs, one for each (primal or dual) cell $\Delta_i$. The sets of vertices of the graph associated with $\Delta_i$ are the set of points whose annuli fully contain $\Delta_i$ and the set of points contained in $\Delta_i$ (one of these sets is a subset of $A$ and the other is a subset of $B$). The total vertex-size over all these graphs is at most $c'(r)(m+n)$, for some constant $c'(r)$ depending on $r$.

We now process each of the $O(r^4)$ subproblems recursively, using the same parameter $r$, and keep doing so until we get subproblems of size at most $L$ (in terms of the number of $A$-points plus the number of $B$-points). If this happens at level $j$ of the recursion, we have (roughly) $(m+n)/r^{3j} = L$, or $r^j = ((m+n)/L)^{1/3}$. The number of subproblems is at most $c^2r^4j = c^2j((m+n)/L)^{4/3}$. If we choose $r$ sufficiently large, we can bound $c^2j$ by $(r^j)^{\varepsilon}$, where $\varepsilon$ is the positive parameter prespecified in the lemma. The total (vertex) size of the graphs output so far is dominated by the size of the graphs output at the last level, which is at most

\[c^2j c'(r)(m+n)/r^{3j} \leq c'(r)(m+n)r^{j(1+\varepsilon)} \leq c'(r)(m+n)^{4/3+\varepsilon}/L^{1/3}.
\]

With some care (again, see [12, 13, 23]), this also bounds the cost of constructing the structure.

We count (within the same time bound) the number $N_1$ of edges in the graphs produced by the algorithm. If $N_1$ exceeds $L/2$, we generate a random sample $R_1$ of $c_1 \log(m+n)$ pairs of points (i.e., edges) from these graphs, for some sufficiently large constant $c_1 > 0$. We omit the routine details of the sampling mechanism, but remark that it ensures that each of the sampled distances is a random element of the (uniform distribution on the) set of all distances recorded by the graphs. By construction, all sampled distances lie in $(\alpha, \beta)$. Moreover, with high probability, the sample contains an approximate median (in the middle half) of these distances; the routine justification of this claim is provided below.

Each subproblem at the bottom of the recursion may contain additional distances that lie in $(\alpha, \beta)$; these are distances between centers of annuli whose boundaries cross the cell of the subproblem and points in

---

1 The construction in [12, 13, 23] shows that each $\Delta_i$ is crossed by at most $m/r$ circles in $C$. To ensure that each $\Delta_i$ contains at most $n/r^2$ points of $B$, we duplicate each $\Delta_i$, that contains more than $n/r^2$ points as many times as needed, and assign to each copy a subset of at most $n/r^2$ of the points (these sets are pairwise disjoint and cover all the points in the cell). Then each cell of the resulting subdivision contains at most $n/r^2$ points, and the size of the cutting is still $O(r^2)$. 

7
the cell that lie inside these annuli. Denote by $N_2$ the overall number of these distances at the bottom subproblems ($N_2$ is not known to us). Note that the total number of pairs in the subproblems is
\[ M := O(( (m + n)/L)^{4/3} \cdot L^2) = O(( m + n)^{4/3 + \varepsilon} L^{2/3}). \]

Our next step is to determine (approximately) how many of these distances lie inside $(\alpha, \beta)$; that is, our goal is to estimate $N_2$. To this end, we generate a random sample $R_2$ of $c_2(M/L) \log(m + n)$ pairs from these subproblems, for some sufficiently large constant $c_2 > 0$, and check how many of them lie in $(\alpha, \beta)$. (Again, the sampling mechanism is straightforward, and we omit its details.)

Let $R'_2$ denote the subset of $R_2$ of those pairs whose distances lie in $(\alpha, \beta)$. It can be shown, similar to the analysis of Har-Peled and Sharir [18] (which in turn is based on the work of Li et al. [22]) that, with high probability, $N_2$ is at most $L/2$ if and only if the number of distances in $R'_2$ is $O(\log(m + n))$, for an appropriate constant of proportionality. (Sharir and Shaul [23], who also use this tool, call such samples shallow $\varepsilon$-nets.)

If $N_1 \leq L/2$ and we have determined that $N_2 \leq L/2$ too, then $N_1 + N_2$ is at most $L$, and we terminate the algorithm (for the task considered in the lemma), since we have determined that the number of distances of $A \times B$ that lie in $(\alpha, \beta)$ is at most $L$. Otherwise, with high probability, $R_1 \cup R_2$ contains an approximate median (in the middle three quarters) of the pairwise distances of $A \times B$ in $(\alpha, \beta)$. In more detail, in the case under consideration we either have $N_1 > L/2$ or $N_2 > L/2$ (or both). In the former case, the probability that $R_1$ does not contain a pair of distances in the middle half of the distances recorded in the complete bipartite graphs is $(1/2)^c_1 \log(m + n) = 1/(m + n)^{c_1}$. If $N_2 > L/2$, then the probability that $R_2$ does not contain a pair at distance in the middle half of the corresponding $N_2$ distances is
\[ \left(1 - \frac{N_2}{2M}\right) c_2(M/L) \log(m + n) < e^{-\frac{1}{2} c_2(N_2/L) \log(m + n)} < e^{-\frac{1}{2} c_2 \log(m + n)} < \frac{1}{(m + n)^{c_2/4}}. \]

Assume for the moment that both $N_1$ and $N_2$ are greater than $L/2$. Then, with high probability, $R_1$ contains a pair $(a'_1, b'_1)$ whose distance, $d_1$, lies in the middle half of the distances recorded in the graphs. Similarly, $R'_2$ contains a pair $(a'_2, b'_2)$ whose distance, $d_2$, lies in the middle half of the distances in $(\alpha, \beta)$ that are generated by the leaf subproblems. An easy calculation shows that either $d_1$ or $d_2$ must lie in the middle three quarters of the overall set of distances in $(\alpha, \beta)$. Similar reasoning applies when either only $N_1$ or only $N_2$ is greater than $L/2$. We thus return $R_1 \cup R'_2$ as the output of the algorithm.

The cost of the algorithm is composed from the following sub-costs:
(i) We pay $O((m + n)^{4/3 + \varepsilon}/L^{1/3})$ for the partial construction of the data structure.
(ii) We then sample and test
\[ O((M/L) \log(m + n)) = O\left( ((m + n)^{4/3 + \varepsilon} L^{2/3}/L) \log(m + n) \right) = O\left( (m + n)^{4/3 + \varepsilon}/L^{1/3} \right) \]
pairs in the bottom subproblems, for another, but still arbitrarily small $\varepsilon > 0$, at the same asymptotic cost.

The way it is described, the algorithm does not verify that the samples that it returns satisfy the desired properties, nor does it verify that the number of distances in $(\alpha, \beta)$ is indeed at most $L$, when it makes this assertion. As such, the running time is deterministic, and the algorithm succeeds with high probability (which can be calibrated by the choice of the constants $c_1, c_2$). See below for another comment regarding this issue.

We use the procedure provided by Lemma 4.1 to find an interval $(\alpha, \beta)$ that contains at most $L$ distances between pairs of $A \times B$, including $\overline{\delta_F(A, B)}$. We find this interval using binary search, starting with
\((\alpha, \beta) = (0, \infty), \) say. In each step of the search, we run the algorithm of Lemma 4.1. If it determines that the number of critical distances in \((\alpha, \beta)\) is at most \(L\) we stop. (The concrete choice of \(L\) that we will use is given later.) Otherwise, the algorithm returns a random sample \(R\) that contains, with high probability, an approximate median (in the middle three quarters) of the distances in \((\alpha, \beta)\). We then find two consecutive distances \(\alpha', \beta'\) in \(R\) such that \(\delta_F(A, B) \in (\alpha', \beta']\), using the decision procedure (see Figure 2). \((\alpha', \beta']\) is a subinterval of \((\alpha, \beta)\) that contains, with high probability, at most \(7/8\) of the distances in \((\alpha, \beta)\). We then proceed to the next step of the binary search, applying again the algorithm of Lemma 4.1 to the new interval \((\alpha', \beta']\). The resulting algorithm runs in \(O((m+n)^{4/3+\varepsilon}/L^{1/3} + (m+n) \log(m+n))\) time, for any \(\varepsilon > 0\).

Once we have narrowed down the interval \((\alpha, \beta)\), so that it now contains at most \(L\) distances between pairs of \(A \times B\), including \(\delta_F(A, B)\), we can find \(\delta_F(A, B)\) by simulating the execution of the decision procedure at the unknown \(\delta_F(A, B)\). A simple way of doing this is as follows. To determine whether \(M_{i,j} = 1\) at \(\delta_F(A, B)\), we compute the critical distance \(r' = \|a_i - b_j\|\) at which \(M_{i,j}\) becomes 1. If \(r' \leq \alpha\) then \(M_{i,j} = 0\), and if \(r' \geq \beta\) then \(M_{i,j} = 1\). Otherwise, \(\alpha < r' < \beta\) is one of the at most \(L\) distances in \((\alpha, \beta)\). In this case we run the decision procedure at \(r'\) to determine \(M_{i,j}\). Since there are at most \(L\) distances in \((\alpha, \beta)\), the total running time is \(O(L(m+n))\). By picking \(L = (m+n)^{1/4+\varepsilon}\) for another, but still arbitrarily small \(\varepsilon > 0\), we balance the bounds of \(O((m+n)^{4/3+\varepsilon}/L^{1/3} + (m+n) \log(m+n))\) and \(O(L(m+n))\), and obtain the bound of \(O((m+n)^{5/4+\varepsilon})\), for any \(\varepsilon > 0\), on the overall running time.

Although this significantly improves the naive implementation mentioned earlier, it suffers from the weakness that it has to run the decision procedure separately for each distance in \((\alpha, \beta)\) that we encounter during the simulation. Lemma 4.2 shows how to accumulate several unknown distances and resolve them all using a binary search that is guided by the decision procedure. This allows us to find \(\delta_F(A, B)\) within the interval \((\alpha, \beta)\) more efficiently.

**Lemma 4.2.** Given a set \(A\) of \(m\) points and a set \(B\) of \(n\) points in the plane, and an interval \((\alpha, \beta) \subset \mathbb{R}\) that contains at most \(L\) distances between pairs in \(A \times B\), including \(\delta_F(A, B)\), we can find \(\delta_F(A, B)\) in \(O((m+n)L^{1/2} \log(m+n))\) (deterministic) time using \(O(m+n)\) space.

**Proof.** We simulate the decision procedure (of Figure 2) at the unknown value \(\delta^- = \delta_F(A, B)\). During the simulation, when attempting to retrieve specific entries \(M_{i,j}\) of \(M\), we encounter comparisons between \(\delta^-\) and concrete distances between pairs of points in \(A \times B\). When we need to compare \(\delta^-\) with such a distance \(r'\), we first check whether \(r'\) is in \((\alpha, \beta)\). If not, we know the result of the comparison (if \(r' \leq \alpha\) then \(r' < \delta^-\), and if \(r' > \beta\) then \(r' > \delta^-\)). If \(\alpha < r' \leq \beta\), we bifurcate, continuing along two possible paths, one assuming that \(r' \leq \delta^-\) and one assuming that \(r' > \delta^-\). However, we proceed along each of these paths for only \(s\) steps, for another parameter \(s\) that will be specified shortly. (More precisely, we proceed until we have examined \(s\) known entries of \(M\) (i.e., entries lying outside \((\alpha, \beta)\)), including 0-entries that we encounter as we climb upwards in a column.)

If, before examining \(s\) entries, we encounter another unknown entry, we bifurcate again, and keep doing so, until we have examined a total of \(m+n\) entries of \(M\), in which case we terminate the current “phase”. (It is conceivable that some entries of \(M\) are examined more than once in this procedure, but when such a multiply-visited entry is unknown, we bifurcate there only once.) The resulting object is a binary tree \(T\), with some number, \(x\), of outdegree-2 nodes (at which we have bifurcated)\(^2\) so that the maximum stretch of consecutive outdegree-1 nodes is \(s\), and so that the total number of nodes in the tree is at most \(m+n\).

We now sort the set \(X\) of the \(x = O(m+n)\) critical values at which we have bifurcated, in \(O((m+n) \log(m+n))\) time. We then run a binary search over \(X\), using the decision procedure (of Figure 2) to

\(^2\)Technically, when we are at some known entry \(M_{i,j}\), check its right neighbor \(M_{i,j+1}\), and find that it is (known to be) 0, we continue the tracing upwards in column \(j\). This 2-way exploration is not considered a bifurcation in the present analysis.
guide the search. This step also takes $O((m + n) \log(m + n))$ time. This determines all the $x$ unknown values that we have encountered, and allows us to choose the lowest path in $T$ that is still a semi-sparse staircase, as the next portion of the overall lowest semi-sparse path $S$ in $M$ (at the optimal value $\delta^-$). This also allows us to shrink the interval that is known to contain $\delta^-$ to be bounded by two consecutive critical values of $\{\alpha, \beta\} \cup X$. What we have gained, in a “successful” phase, is at least $s$ extra steps of the desired semi-sparse path $S$. Assuming this to be the case, and since the total length of $S$ is $O(m + n)$, we need at most $O((m + n)/s)$ successful phases of this kind, whose total cost is thus $O(((m + n)^2/s) \log(m + n))$.

This is only one side of the story, though, because there might be phases where we do not manage to gain $s$ steps, because we run into “too many” bifurcations. If a phase generates $x$ bifurcations, then, continuing the search in $M$ beyond them, we encounter at most $xs$ entries of $M$. The reason for not having a “tail” of $s$ entries beyond any bifurcation is that we have exceeded the number of steps per phase, namely $m + n$. We thus have $m + n \leq xs$ or $x \geq (m + n)/s$. In other words, the number of such “unsuccessful” phases is $O(Ls/(m + n))$, and each such phase takes $O((m + n) \log(m + n))$ time, as before, for a total of $O(Ls \log(m + n))$ time. At the end of the simulation, we have two consecutive critical values $\alpha, \beta$ of distances between pairs of $A \times B$, where $\alpha < \delta^- \leq \beta$, so we conclude that $\delta^-(A, B) = \delta^- = \beta$.

Overall, the cost is

$$O\left(\frac{(m + n)^2 \log(m + n)}{s} + Ls \log(m + n)\right),$$

and we make the overall cost of this stage $O\left((m + n)L^{1/2}\log(m + n)\right)$, by choosing $s = (m + n)/L^{1/2}$ to balance the terms.

Note that after each phase of the algorithm, we can free the memory used to process the phase and only remember $\alpha, \beta$ and the path in $M$ (a prefix of the desired $S$) that we have traversed so far. Since each phase processes $O(m + n)$ entries of $M$, the space needed by this algorithm is $O(m + n)$. \hfill \Box

To balance the two terms in Lemma 4.1 and Lemma 4.2, we choose $L = (m + n)^{2/5+\varepsilon}$, for another, but still arbitrarily small $\varepsilon > 0$. This gives the following main result of this section.

**Theorem 4.3.** Given a set $A$ of $m$ points and a set $B$ of $n$ points in the plane, and a parameter $\varepsilon > 0$, we can compute the one-sided discrete Fréchet distance $\delta^-(A, B)$ with shortcuts in randomized expected time $O((m + n)^{6/(5+\varepsilon)} \log(m + n))$ using $O((m + n)^{6/(5+\varepsilon)})$ space.

**Proof.** All the details of the proof have already been given, except for the precise statement concerning the running time. As noted earlier, the algorithm of Lemma 4.1 does not verify explicitly that the sample that it generates does contain an approximate median, nor does it verify that the number of distances in $(\alpha, \beta]$ is at most $L$ when it so asserts.

We can either let things stay as they are, knowing that the algorithm will succeed with overall high probability. Alternatively, as asserted in the theorem, we can realize that something went wrong in one of two situations: Either we apply the algorithm of Lemma 4.1 too many times, or we encounter a total of more than $L$ bifurcations during the execution of the algorithm of Lemma 4.2. In these cases we scrap the whole execution and start afresh from scratch. The expected number of rounds of this kind is $O(1)$, and the theorem follows. \hfill \Box

**Remark.** In principle, our algorithm for the one-sided Fréchet distance with shortcuts can be generalized to higher dimensions. The only part that limits our approach to $\mathbb{R}^2$ is the algorithm of Lemma 4.1. However, this part can be replaced by a random sampling approach that is similar to the one that we use in Lemma 6.5.
for the semi-continuous Fréchet distance with shortcuts. This will increase the running time of the algorithm, but it will stay strictly subquadratic.

We believe that Lemma 4.1 is of independent interest, and that it may find other applications in distance-related optimization problems. Here is one such example.

**Corollary 4.4.** Given a set \( A \) of \( m \) points and a set \( B \) of \( n \) points in the plane, and parameters \( 0 < k < mn \), \( 0 < L < k \), and \( \varepsilon > 0 \), we can find a pair \((a, b)\) \( \in A \times B \) such that, with high probability, \( \|a - b\| \) is the \( k \)th smallest distance between a point of \( A \) and a point of \( B \), for some rank \( \kappa \) satisfying \( k - L < \kappa < k + L \), in \( O((m + n)^{4/3+\varepsilon}/L^{1/3} + m + n) \) time, using \( O((m + n)^{4/3+\varepsilon}/L^{1/3} + m + n) \) space.

**Proof.** Consider the following decision problem (already discussed earlier). Given a set \( A \) of \( m \) points, a set \( B \) of \( n \) points, a parameter \( \delta > 0 \) and a parameter \( 0 < k < mn \), determine whether the number, \( N \), of pairs in \( A \times B \) at distance at most \( \delta \) is at most \( k \). To solve this decision problem, we use the algorithm of Lemma 4.1 with \( k \) as its parameter \( L \), but we replace the annuli centered at the points of \( A \) and \( B \) by respective disks of radius \( \delta \) centered at the same points. Since a point \( a \in A \) is at distance at most \( \delta \) from a point \( b \in B \) if and only if \( a \) is in the disk of radius \( \delta \) centered at \( b \), and vice versa, the algorithm allows us to determine, with high probability, whether \( N \) is at most \( k \). The cost of this step is \( O((m + n)^{4/3+\varepsilon}/k^{1/3} + m + n) \), and it is subsumed by the cost of the further steps.

Let \( \delta^k(A, B) \) denote the \( k \)th smallest distance between a point of \( A \) and a point of \( B \). We now use again the algorithm of Lemma 4.1, together with the above decision procedure, to find an interval \((\alpha, \beta]\) that contains at most \( L \) pairwise distances from \( A \times B \), including \( \delta^k(A, B) \). To this end, we repeatedly shrink \((\alpha, \beta]\) using a binary search, starting with \((\alpha, \beta]\) in \( (0, \infty) \), say. In each step of the search, we call the algorithm of Lemma 4.1 (this time, in its original setup, with \( L \) as the parameter). If it determines that the number of critical distances in \((\alpha, \beta]\) is at most \( L \), we output \( \alpha \) (together with its generating pair) as an approximation for \( \delta^k(A, B) \), in the sense asserted in the lemma (\( \beta \) would do equally well). Otherwise, we have a random sample \( R \) that contains, with high probability, an approximate median (in the middle three quarters) of the pairwise distances in \((\alpha, \beta]\). We locate a consecutive pair \( x, y \) of distances in \( R \), using the decision procedure, such that the interval \([x, y]\) contains \( \delta^k(A, B) \). Since \( R \) contains an approximate median, the number of distances in \([x, y]\) is, with high probability, at most \( 7/8 \) of the number of distances in \((\alpha, \beta]\). We then proceed with the next step of the search. The overall resulting algorithm runs in \( O((m + n)^{4/3+\varepsilon}/L^{1/3} + m + n) \) time, and it uses \( O((m + n)^{4/3+\varepsilon}/L^{1/3} + m + n) \) space. \( \Box \)

**Remark.** This should be compared with the near-linear algorithms in \([5, 17]\) that approximate the value of the \( k \)th smallest distance in \( A \times B \) (rather than its rank as provided in Corollary 4.4). It would be interesting to understand better the relationship between these algorithms and ours.

### 5 The two-sided DFDS

We first consider the corresponding decision problem. That is, given \( \delta > 0 \), we wish to decide whether \( \delta^+_F(A, B) \leq \delta \).

Consider the matrix \( M \) as defined in Section 5. In the two-sided version of DFDS, given a reachable position \((a_i, b_j)\) of the frogs, the \( A \)-frog can make a *skipping upward move*, as in the one-sided variant, to any point \( a_k, k > i \), for which \( M_{k,j} = 1 \). Alternatively, the \( B \)-frog can jump to any point \( b_l, l > j \), for which \( M_{i,l} = 1 \); this is a *skipping right move* in \( M \) from \( M_{i,j} = 1 \) to \( M_{i,l} = 1 \), defined analogously. Determining whether \( \delta^+_F(A, B) \leq \delta \) corresponds to deciding whether there exists a *sparse staircase* of ones
in $M$ that starts at $M_{1,1}$, ends at $M_{m,n}$, and consists of an interweaving sequence of skipping upward moves and skipping right moves (see Figure 1(c)).

Katz and Sharir [19] showed that the set $S = \{(a_i, b_j) | \|a_i - b_j\| \leq \delta\} = \{(a_i, b_j) | M_{i,j} = 1\}$ can be computed, in $O((m^{2/3}n^{2/3} + m + n)\log n)$ time and space, as the union of the edge sets of a collection $\Gamma = \{A_t \times B_t | A_t \subseteq A, B_t \subseteq B\}$ of edge-disjoint complete bipartite graphs. The number of graphs in $\Gamma$ is $O(m^{2/3}n^{2/3} + m + n)$, and the overall sizes of their vertex sets are

$$\sum_t |A_t|, \sum_t |B_t| = O((m^{2/3}n^{2/3} + m + n)\log n).$$

We store each graph $A_t \times B_t \in \Gamma$ as a pair of sorted linked lists $L_{A_t}$ and $L_{B_t}$ over the points of $A_t$ and of $B_t$, respectively. For each graph $A_t \times B_t \in \Gamma$, there is 1 in each entry $M_{i,j}$ such that $(a_i, b_j) \in A_t \times B_t$.

That is, $A_t \times B_t$ corresponds to a submatrix $M^{(t)}$ of ones in $M$ (whose rows and columns are not necessarily consecutive). See Figure 3(a).

Note that if $(a_i, b_j) \in A_t \times B_t$ is a reachable position of the frogs, then every pair in the set $\{(a_k, b_l) \in A_t \times B_t | k \geq i, l \geq j\}$ is also a reachable position. (In other words, the positions in the upper-right submatrix of $M^{(t)}$ whose lower-left entry is $M_{i,j}$ are all reachable; see Figure 3(b)).

![Figure 3: (a) A possible representation of the matrix $M$ as a collection of submatrices of ones, corresponding to the complete bipartite graphs $\{a_1, a_2\} \times \{b_1, b_2\}$, $\{a_1, a_3, a_5\} \times \{b_4, b_6\}$, $\{a_1, a_3\} \times \{b_7, b_{11}\}$, $\{a_2, a_3, a_5\} \times \{b_5, b_8, b_9\}, \{a_4, a_7, a_8\} \times \{b_3, b_4\}, \{a_4, a_7\} \times \{b_6, b_{10}\}, \{a_6\} \times \{b_9, b_{11}\}, \{a_8\} \times \{b_9, b_{12}\}$. (b) Another matrix $M$, similarly decomposed, where the reachable positions are marked with an x.](image)

We say that a graph $A_t \times B_t \in \Gamma$ intersects a row $i$ (resp., a column $j$) in $M$ if $a_i \in A_t$ (resp., $b_j \in B_t$). We denote the subset of graphs of $\Gamma$ that intersect the $i$th row of $M$ by $\Gamma_i^c$ and those that intersect the $j$th column by $\Gamma_j^c$. The sets $\Gamma_i^c$ are easily constructed from the lists $L_{A_t}$ of the graphs in $\Gamma$, and are maintained as linked lists. Similarly, the sets $\Gamma_j^c$ are constructed from the lists $L_{B_t}$, and are maintained as doubly-linked lists, so as to facilitate deletions of elements from them. We have $\sum_i |\Gamma_i^c| = \sum_t |A_t| = O((m^{2/3}n^{2/3} + m + n)\log n)$ and $\sum_j |\Gamma_j^c| = \sum_t |B_t| = O((m^{2/3}n^{2/3} + m + n)\log n)$.

We define a 1-entry $(a_k, b_j)$ to be reachable from below row $i$, if $k \geq i$ and there exists an entry $(a_l, b_j)$, $l < i$, which is reachable. We process the rows of $M$ in increasing order and for each graph $A_t \times B_t \in \Gamma$ maintain a reachability variable $v_t$, which is initially set to $\infty$. We maintain the invariant that when we start processing row $i$, if $A_t \times B_t$ intersects at least one row that is not below the $i$th row, then $v_t$ stores the smallest index $j$ for which there exists an entry $(a_k, b_j) \in A_t \times B_t$ that is reachable from below row $i$.

Before we start processing the rows of $M$, we verify that $M_{1,1} = 1$ and $M_{m,n} = 1$, and abort the computation if this is not the case, determining that $\delta^*_\Gamma(A, B) > \delta$.

Assuming that $M_{1,1} = 1$, each position in $P_1 = \{(a_1, b_1) | M_{1,1} = 1\}$ is a reachable position. It follows that for each graph $A_t \times B_t \in \Gamma$, $v_t$ should be set to $\min\{l | A_t \times B_t \in \Gamma_i^c\}$ and $(a_1, b_1) \in P_1$. Note that graphs $A_t \times B_t$ in this set are not necessarily in $\Gamma_i^c$. We update the $v_t$’s using this rule, as follows. We
first compute $P_1$, the set of pairs, each consisting of $a_1$ and an element of the union of the lists $L_{B_1}$, for $A_t \times B_t \in \Gamma_1^r$. Then, for each $(a_1, b_1) \in P_1$, we set, for each graph $A_u \times B_u \in \Gamma_1^r$, $v_u \leftarrow \min\{v_u, l\}$.

In principle, this step should now be repeated for each row $i$. That is, we should compute $y_i = \min\{v_i \mid A_t \times B_t \in \Gamma_i^r\}$; this is the index of the leftmost entry of row $i$ that is reachable from below row $i$. Next, we should compute $P_i = \{(a_i, b_i) \mid m, l = 1$ and $l \geq y_i\}$ as the union of those pairs that consist of $a_i$ and an element of

$$\{b_j \mid b_j \in L_{B_i} \text{ for } A_t \times B_t \in \Gamma_i^r \text{ and } j \geq y_i\}.$$

The set $P_i$ is the set of reachable positions in row $i$. Then we should set for each $(a_1, b_1) \in P_i$ and for each graph $A_u \times B_u \in \Gamma_i^r$, $v_u \leftarrow \min\{v_u, l\}$. This is too expensive, because it may make us construct explicitly all the 1-entries of $M$.

To reduce the cost of this step, we note that, for any graph $A_t \times B_t$, as soon as $v_t$ is set to some column $l$ at some point during processing, we can remove $b_l$ from $L_{B_t}$, because its presence in this list has no effect on further updates of the $v_t$’s. Hence, at each step in which we examine a graph $A_t \times B_t \in \Gamma_i^r$, for some column $l$, we remove $b_l$ from $L_{B_t}$. This removes $b_l$ from any further consideration in rows with index greater than $i$ and, in particular, $\Gamma_i^r$ will not be accessed anymore. This is done also when processing the first row.

Specifically, we process the rows in increasing order and when we process row $i$, we first compute $y_i = \min\{v_i \mid A_t \times B_t \in \Gamma_i^r\}$, in a straightforward manner. (If $i = 1$, then we simply set $y_1 = 1$.) Then we construct a set $P'_i \subseteq P_i$ of the “relevant” (i.e., reachable) 1-entries in the $i$-th row as follows. For each graph $A_t \times B_t \in \Gamma_i^r$ we traverse (the current) $L_{B_t}$ backwards, and for each $b_j \in L_{B_t}$ such that $j \geq y_i$ we add $(a_i, b_j)$ to $P'_i$. Then, for each $(a_i, b_l) \in P'_i$, we go over all graphs $A_u \times B_u \in \Gamma_i^r$, and set $v_u \leftarrow \min\{v_u, l\}$. After doing so, we remove $b_l$ from all the corresponding lists $L_{B_u}$.

When we process row $m$ (the last row of $M$), we set $y_m = \min\{v_i \mid A_t \times B_t \in \Gamma_m^r\}$. If $y_m < \infty$, we conclude that $\delta_F^+(A, B) \leq \delta$ (recalling that we already know that $M_{m,n} = 1$). Otherwise, we conclude that $\delta_F^+(A, B) > \delta$.

**Correctness.** We need to show that $\delta_F^+(A, B) \leq \delta$ if and only if $y_m < \infty$ (when we start processing row $m$). To this end, we establish in Lemma 5.1 that the invariant stated above regarding $v_t$ indeed holds. Hence, if $y_m < \infty$, then the position $(a_m, b_m)$ is reachable from below row $m$, implying that $(a_m, b_n)$ is also a reachable position and thus $\delta_F^+(A, B) \leq \delta$. Conversely, if $\delta_F^+(A, B) \leq \delta$ then $(a_m, b_n)$ is a reachable position. So, either $(a_m, b_n)$ is reachable from below row $m$, or there exists a position $(a_m, b_j)$, $j < n$, that is reachable from below row $m$ (or both). In either case there exists a graph $A_t \times B_t$ in $\Gamma_m^r$ such that $v_t \leq n$ and thus $y_m < \infty$. We next show that the reachability variables $v_t$ of the graphs in $\Gamma$ are maintained correctly.

**Lemma 5.1.** For each $i = 1, \ldots, m$, the following property holds. Let $A_t \times B_t$ be a graph in $\Gamma_i^r$, and let $j$ denote the smallest index for which $(a_i, b_j) \in A_t \times B_t$ and $(a_i, b_j)$ is reachable from below row $i$. Then, when we start processing row $i$, we have $v_t = j$.

**Proof.** We prove this claim by induction on $i$. For $i = 1$, this claim holds trivially. We assume then that $i > 1$ and that the claim is true for each row $i' < i$, and show that it also holds for row $i$.

Let $A_t \times B_t$ be a graph in $\Gamma_i^r$, and let $j$ denote the smallest index for which there exists a position $(a_i, b_j) \in A_t \times B_t$ that is reachable from below row $i$. We need to show that $v_t = j$ when we start processing row $i$.

Since $(a_i, b_j)$ is reachable from below row $i$, there exists a position $(a_k, b_j)$, with $k < i$, that is reachable, and we let $k_0$ denote the smallest index for which $(a_{k_0}, b_j)$ is reachable. Let $A_o \times B_o$ be the graph containing $(a_{k_0}, b_j)$. We first claim that when we start processing row $k_0$, $b_j$ was not yet deleted from $L_{B_o}$ (nor from
the corresponding list of any other graph in \( \Gamma_j \). Assume to the contrary that \( b_j \) was deleted from \( L_{B_0} \) before processing row \( k_0 \). Then there exists a row \( z < k_0 \) such that \((a_z, b_j) \in P_z' \) and we deleted \( b_j \) from \( L_{B_0} \) when we processed row \( z \). By the last assumption, \((a_z, b_j) \) is a reachable position. This is a contradiction to \( k_0 \) being the smallest index for which \((a_{k_0}, b_j) \) is reachable. (The same argument applies for any other graph, instead of \( A_o \times B_o \).)

We next show that \( v_t \leq j \). Since \((a_{k_0}, b_j) \in A_o \times B_o, A_o \times B_o \in \Gamma_{k_0}^r \cap \Gamma_j \). Since \( k_0 \) is the smallest index for which \((a_{k_0}, b_j) \) is reachable, there exists an index \( j_0 \), such that \( j_0 < j \) and \((a_{k_0}, b_{j_0}) \) is reachable from below row \( k_0 \). (If \( k_0 = 1 \), we use instead the starting placement \((a_1, b_1) \).) It follows from the induction hypothesis that \( y_{k_0} \leq j_0 < j \). Thus, when we processed row \( k_0 \) and we went over \( L_{B_o} \), we encountered \( b_j \) (as just argued, \( b_j \) was still in that list), and we consequently updated the reachability variables \( v_u \) of each graph \( \Gamma_j \), including our graph \( A_t \times B_t \) to be at most \( j \).

(Not that if there is no position in \( A_t \times B_t \) that is reachable from below row \( i \) (i.e., \( j = \infty \)), we trivially have \( v_t \leq \infty \).)

Finally, we show that \( v_t = j \). Assume to the contrary that \( v_t = j_1 < j \) when we start processing row \( i \). Then we have updated \( v_t \) to hold \( j_1 \) when we processed \( b_{j_1} \) at some row \( k_1 < i \). So, by the induction hypothesis, \( y_{k_1} \leq j_1 \), and thus \((a_{k_1}, b_{j_1}) \) is a reachable position. Moreover, \( A_t \times B_t \in \Gamma_j \), since \( v_t \) has been updated to hold \( j_1 \) when we processed \( b_{j_1} \). It follows that \((a_{i}, b_{j_1}) \in A_t \times B_t \). Hence, \((a_{i}, b_{j_1}) \) is reachable from below row \( i \). This is a contradiction to \( j \) being the smallest index such that \((a_{i}, b_{j}) \) is reachable from below row \( i \). This establishes the induction step and thus completes the proof of the lemma.

**Running Time.** We first analyze the initialization cost of the data structure, and then the cost of traversal of the rows for maintaining the variables \( v_t \).

**Initialization.** Constructing \( \Gamma \) takes \( O((m^{2/3}n^{2/3} + m + n) \log(m + n)) \) time. Sorting the lists \( L_{A_t} \) (resp., \( L_{B_t} \)) of each \( A_t \times B_t \in \Gamma \) takes \( O((m^{2/3}n^{2/3} + m + n) \log^2(m + n)) \) time. Constructing the lists \( \Gamma_r^r \) (resp., \( \Gamma_c^c \)) for each \( a_i \in A \) (resp., \( b_j \in B \)) takes time linear in the sum of the sizes of the \( A_i \)'s and the \( B_j \)'s, which is \( O((m^{2/3}n^{2/3} + m + n) \log(m + n)) \).

**Traversing the rows.** When we process row \( i \) we first compute \( y_i \) by scanning \( \Gamma_i^r \). This takes a total of \( O(\sum_i |\Gamma_i^r|) = O((m^{2/3}n^{2/3} + m + n) \log n) \) for all rows. Since the lists \( L_{B_t} \) are sorted, the computation of \( P'_i \) is linear in the size of \( P'_i \). This is so because, once we have added a pair \((a_i, b_j) \) to \( P'_i \), we remove \( b_j \) from all lists that contain it, so we will not encounter it again when scanning other lists \( L_{B_t} \). For each pair \((a_i, b_j) \) in \( P'_i \) we scan \( \Gamma_i^r \), which must contain at least one graph \( A_t \times B_t \in \Gamma \) such that \( a_i \in A_t \) (and \( b_j \in B_t \)). For each element \( A_t \times B_t \in \Gamma_i^r \) we spend constant time updating \( v_t \) and removing \( b_j \) from \( L_{B_t} \). It follows that the total time, over all rows, of computing \( P'_i \) and scanning the lists \( \Gamma_i^r \) is \( O(\sum_i |\Gamma_i^r|) = O((m^{2/3}n^{2/3} + m + n) \log n) \).

We conclude that the total running time is \( O((m^{2/3}n^{2/3} + m + n) \log^2(m + n)) \).

**The optimization procedure.** We use the above decision procedure for finding the optimum \( \delta^*_F(A, B) \), as follows. Note that if we increase \( \delta \) continuously, the set of 1-entries of \( M \) can only grow, and this can only happen at a distance between a point of \( A \) and a point of \( B \). We thus perform a binary search over the \( mn \) pairwise distances between the pairs of \( A \times B \). In each step of the search we need to determine the \( k \)th smallest pairwise distance \( r_k \) in \( A \times B \), for some value of \( k \). We do so by using the distance selection algorithm of Katz and Sharir [19], which can easily be adapted to work for this bichromatic scenario. We then run the decision procedure on \( r_k \), using its output to guide the binary search. At the end of this search, we obtain two consecutive critical distances \( \delta_1, \delta_2 \) such that \( \delta_1 < \delta^*_F(A, B) \leq \delta_2 \), and we can therefore conclude that \( \delta^*_F(A, B) = \delta_2 \). The running time of the distance selection algorithm of [19] is
Given a set \( A \) of \( m \) points and a set \( B \) of \( n \) points in the plane, we can compute the two-sided discrete Fréchet distance with shortcuts \( \delta^2_F(A,B) \), in time \( O((m^{2/3}n^{2/3} + m + n) \log^3(m + n)) \), using \( O((m^{2/3}n^{2/3} + m + n) \log(m + n)) \) space.

### 6 Semi-continuous Fréchet distance with shortcuts

Let \( f \subseteq \mathbb{R}^2 \) denote a polygonal curve with \( n \) edges \( e_1, \ldots, e_n \) and \( n + 1 \) vertices \( p_0, p_1, \ldots, p_n \), and let \( A = (a_1, \ldots, a_m) \) denote a sequence of \( m \) points in the plane. Consider a person that is walking along \( f \) from its starting endpoint to its final endpoint, and a frog that is jumping along the sequence \( A \) of stones. The frog is allowed to make shortcuts (i.e., skip stones) as long as it traverses \( A \) in the right (increasing) direction, but the person must trace the complete curve \( f \) (see Figure 4(a)). Assuming that the person holds the frog by a leash, our goal is to compute the minimal length \( \delta^s_F(A,f) \) of a leash that is required in order to traverse \( f \) and (parts of) \( A \) in this manner, taking the frog and the person from \((a_1, p_0)\) to \((a_m, p_n)\).

![Figure 4](image)

Figure 4: (a) A curve \( f \) and a sequence of points \( A = (a_1, \ldots, a_5) \). (b) Thinking of \( f \) as a continuous mapping from \([0,1]\) to \( \mathbb{R}^2 \), the \( i \)th row depicts the set \( \{ t \in [0,1] \mid f(t) \in D_\delta(a_i) \} \). The dotted subintervals and their connecting upward moves (not drawn) constitute the lowest semi-sparse staircase between the starting and final positions.

Consider the decision version of this problem, where, given a parameter \( \delta > 0 \), we wish to decide whether the person and the frog can traverse \( f \) and (parts of) \( A \) using a leash of length \( \delta \). This problem can be solved using the algorithm for solving the one-sided DFDS, with a slight modification that takes into account the continuous nature of \( f \). Specifically, for a point \( p \in \mathbb{R}^2 \), let \( D_\delta(p) \) denote the disk of radius \( \delta \) centered at \( p \). Now, consider a vector \( M \) whose entries correspond to the points of \( A \). For each \( i = 1, \ldots, m \), the \( i \)th entry of \( M \) is

\[
M_i = M(a_i) = f \cap D_\delta(a_i)
\]

(see Figure 4(b)). Each \( M_i \) is a finite union of connected subintervals of \( f \). We do not compute \( M \) explicitly, because the overall “description complexity” of its entries might be too large. Specifically, the number of connected subsegments of the edges of \( f \) that comprise the elements of \( M \) can be \( mn \) in the worst case.

Instead, we assume availability of (efficient implementations of) the following two primitives.

(i) **NextEndpoint**\((x, a_i)\): Given a point \( x \in f \) and a point \( a_i \) of \( A \), such that \( x \in D_\delta(a_i) \), return the forward endpoint of the connected component of \( f \cap D_\delta(a_i) \) that contains \( x \).
(ii) **NextDisk**(x, a<sub>i</sub>): Given x and a<sub>i</sub>, as in (i), find the smallest j > i such that x ∈ D<sub>δ</sub>(a<sub>j</sub>), or report that no such index exists (return j = ∞).

Both primitives admit efficient implementations. For our purposes it is sufficient to implement Primitive (i) by traversing the edges of f one by one, starting from the edge containing x, and checking for each such edge e<sub>j</sub> of f whether the forward endpoint p<sub>j</sub> of e<sub>j</sub> belongs to D<sub>δ</sub>(a<sub>i</sub>). For the first e<sub>j</sub> for which this test fails, we return the forward endpoint of the interval e<sub>j</sub> ∩ D<sub>δ</sub>(a<sub>i</sub>). It is also sufficient to implement Primitive (ii) by checking for each j > i in increasing order, whether x ∈ D<sub>δ</sub>(a<sub>j</sub>), and return the first j for which this holds. To solve the decision problem, we proceed as in the decision procedure of the one-sided DFDS (see Figure 2). More specifically, here a

\[
\begin{align*}
\text{Input: } & A, f, \delta \\
& S \leftarrow \emptyset \\
& a^1 \leftarrow a_1, x^1 \leftarrow p_0 \\
\text{If } (x^1 \notin D_\delta(a^1)) & \text{ then}
\begin{align*}
& \quad \text{Return } \delta_F^e(A, f) > \delta \\
& \text{Add } (a^1, x^1) \text{ to } S \\
& k \leftarrow 1 \\
\text{While } (a^k \text{ is not } a_m \text{ or } f \text{ is not fully traversed}) & \text{ do}
\begin{align*}
& \quad x^{k+1} \leftarrow \text{NextEndpoint}(x^k, a^k) \\
& \quad \text{Add } (a^k, x^{k+1}) \text{ to } S \\
& \quad \text{If } (a^k = a_m \text{ and } x^{k+1} = p_n) \text{ then}
\begin{align*}
& \quad \quad \text{Return } \delta_F^e(A, f) \leq \delta \\
& \quad \quad l \leftarrow \text{NextDisk}(x^{k+1}, a^k) \\
& \quad \text{If } (l \leq n) \text{ then}
\begin{align*}
& \quad \quad \quad a^{k+1} \leftarrow a_l \\
& \quad \quad \quad \text{Add } (a^{k+1}, x^{k+1}) \text{ to } S \\
& \quad \quad \text{Else}
\begin{align*}
& \quad \quad \quad \text{Return } \delta_F^e(A, f) > \delta \\
& \quad \quad \quad k \leftarrow k + 1
\end{align*}
\end{align*}
\end{align*}
\end{align*}
\end{align*}
\end{align*}

Figure 5: The decision procedure Γ for the semi-continuous Fréchet distance with shortcuts.

The path S computed by Γ is a sequence of reachable positions (a<sup>1</sup>, x<sup>1</sup>), (a<sup>2</sup>, x<sup>2</sup>), ..., where a<sup>k</sup> is a point of A and x<sup>k</sup> is a point on an edge of f. Let P be the sequence of pairs (a<sup>1</sup>, s<sup>1</sup>), (a<sup>2</sup>, s<sup>2</sup>), ..., where s<sup>k</sup> is either the edge of f containing x<sup>k</sup> (in its relative interior) or the vertex of f coinciding with x<sup>k</sup>, for k = 1, 2, ... .

The correctness of the decision procedure Γ is proved similarly to the correctness of the decision procedure of the one-sided DFDS (of Figure 2). More specifically, here a *semi-continuous semi-sparse staircase* is an interweaving sequence of *discrete skipping upward moves* and *continuous right moves*, where a discrete skipping upward move is a move from a reachable position (a<sub>i</sub>, p) of the frog and the person to another

---

\[\text{Note that we use superscripts } a^k \text{ and } s^k \text{ to denote the sequence } S \text{ defining the solution produced by the decision procedure. This is to distinguish them from } a_k \text{ and } e_k \text{ or } p_k, \text{ with subscripts, that denote the original input sequence of points of the frog and the sequence of segments and their endpoints on } f.\]
position \((a_j, p)\) such that \(j > i\) and \(p \in D_\delta(a_j)\). A continuous right move is a move from a reachable position \((a_i, p)\) of the frog and the person to another position \((a_i, p')\) where \(p'\), and the entire portion of \(f\) between \(p\) and \(p'\), are contained in \(D_\delta(a_i)\). Then there exists a semi-continuous semi-sparse staircase that reaches the position \((a_m, p_n)\) if and only if \(\delta_F^s(A, f) \leq \delta\).

Concerning correctness, we prove that if there exists a semi-continuous semi-sparse staircase \(S'\) that reaches position \((a_m, p_n)\), then the decision procedure maintains a partial semi-continuous semi-sparse staircase \(S\) that is always “below” \(S'\) (in terms of the corresponding indices of the positions of the frog), and therefore \(S\) reaches a position where the person is at \(p_n\) (and the frog can then jump directly to \(a_m\)). Intuitively, this holds since the decision procedure can at any point join the plot of \(S\) on \(a\) staircase \(S\) and therefore \(\delta\) is at most \(0\), we can determine whether the semi-continuous Fréchet distance \(\delta_F^s(A, f)\) with shortcuts between \(A\) and \(f\) is at most \(\delta\), in \(O(m + n)\) time, using \(O(m + n)\) space.

**Lemma 6.1.** Given a polygonal curve \(f\) with \(n\) edges in the plane, a set \(A\) of \(m\) points in the plane, and a parameter \(\delta > 0\), we can determine whether the semi-continuous Fréchet distance \(\delta_F^s(A, f)\) with shortcuts between \(A\) and \(f\) is at most \(\delta\), in \(O(m + n)\) time, using \(O(m + n)\) space.

**The optimization procedure.** We now use the decision procedure \(\Gamma\) to find the optimal value \(\delta_F^s(A, f)\). To make the dependence on \(\delta\) explicit, we denote, in what follows, the decision procedure for distance \(\delta\) by \(\Gamma(\delta)\). The path \(S\) computed by \(\Gamma(\delta)\), and each element \((a_k, x_k)\) of \(S\), depend on \(\delta\), so we denote them by \(S(\delta), a_k(\delta)\) and \(x_k(\delta)\), respectively. The sequence \(P\) of pairs \((a_k, s_k)\), and each of its elements, also depend on \(\delta\), so we denote \(P\) by \(P(\delta)\), and \(s_k\) by \(s_k(\delta)\). Of course, \(\Gamma(\delta)\) might fail, i.e., report that \(\delta_F^s(A, f) > \delta\). In such a case, the path \(S(\delta)\) and the sequence of pairs \(P(\delta)\) consist of everything that was accumulated in them until \(\Gamma(\delta)\) has terminated (that is, aborted). In particular, \(S(\delta)\) does not end in this case at \((a_m, p_n)\).

The path \(S(\delta_1)\) is combinatorially different from the path \(S(\delta_2)\), for \(\delta_1, \delta_2 > 0\), if \(P(\delta_1) \neq P(\delta_2)\); otherwise, we say that \(S(\delta_1)\) and \(S(\delta_2)\) are combinatorially equivalent.

We next argue that each critical value of \(\delta\) where \(S(\delta)\) changes combinatorially must be of one of the following two types:

1. The distance between a point of \(A\) and a vertex of \(f\) (point-vertex distance).
2. For two points \(p, q \in A\) and an edge \(e\) of \(f\), the distance between \(p\) (or \(q\)) and the intersection of \(e\) with the bisector of \(p\) and \(q\) (point-point-edge distance).

See Figure 9 for an illustration. We assume general position of the input, so as to ensure that these critical distances are all distinct.

**Lemma 6.2.** Let \(\delta\) be such that \(S(\delta^-)\) is combinatorially different from \(S(\delta)\), for all \(\delta^- < \delta\) and arbitrarily close to \(\delta\). Then \(\delta\) is either a point-vertex distance or a point-point-edge distance.

**Proof.** In what follows, we use \(\delta^-\) to denote an arbitrary point from the neighborhood of \(\delta\) mentioned in the lemma. Consider the point at which the executions of \(\Gamma(\delta^-)\) and of \(\Gamma(\delta)\) add a pair to \(P(\delta^-)\) which is different from the pair added to \(P(\delta)\) (this includes the case in which we add a pair to only one of the sets \(P(\delta^-), P(\delta)\)). If \((a_1, p_0)\) is in \(P(\delta)\) but not in \(P(\delta^-)\) then \(\delta\) is the distance between \(a_1\) and \(p_0\), a point-vertex distance. Otherwise, assume that the different pairs arose following a call to \(\text{NextEndPoint}(x_k, a_k)\).
Choosing Lemma 6.3. Lemma (Lemma 4.2) for the discrete case. Raichel (see [17]). The proof of Lemma 6.4 is similar to but more involved than the proof of the analogous

time. Then, Lemma 6.4 shows that we can find

\[ \delta \]

no more than

\[ L \]

for the corresponding disk has no effect on

this is because the appearance or disappearance of the small interval of intersection between the edge and

with a point-point-edge distance, which is ruled out anyway by our general position assumption); informally,

Figure 6: Two of the critical distances between \( f \) and \( A \). \( \delta_1 \) is a point-vertex distance between \( a_4 \) and \( p_2 \). \( \delta_2 \) is a point-point-edge distance between \( a_1 \), \( a_3 \) and \( e_1 \).

Then \( x^{k+1}(\delta) = \text{NextEndPoint}(x^k(\delta), a^k(\delta)) \) and \( x^{k+1}(\delta^-) = \text{NextEndPoint}(x^k(\delta^-), a^k(\delta^-)) \) belong to different edges / vertices of \( f \). Note that \( a^k(\delta) = a^k(\delta^-) \) since this is the first call that causes a discrepancy between \( P(\delta) \) and \( P(\delta^-) \). A simple continuity argument implies that the disk of radius \( \delta \) about \( a^k(\delta) \) must contain a vertex of \( f \), so \( \delta \) is a point-vertex distance.

Finally assume that the first difference in the pairs added to \( P(\delta^-) \) and \( P(\delta) \) occurred following a call to \( \text{NextDisk}(x^{k+1}, a^k) \). Put \( a_\ell(\delta) = \text{NextDisk}(x^{k+1}(\delta), a^k(\delta)) \) and \( a_\ell(\delta^-) = \text{NextDisk}(x^{k+1}(\delta^-), a^k(\delta^-)) \).

As before, \( a^k(\delta) = a^k(\delta^-) \) by our assumption. Moreover, since \( x^{k+1}(\delta) \) is not a vertex of \( f \) (or else the previous call to \( \text{NextEndPoint} \) would have produced different pairs at \( \delta^- \) and at \( \delta \)), a simple continuity argument shows that \( x^{k+1}(\delta^-) \to x^{k+1}(\delta) \) as \( \delta^- \to \delta \). Assume that \( \ell(\delta^-) \neq \ell(\delta) \). We claim that in this case \( x^{k+1}(\delta) \) must lie on \( \partial D_\delta(a_\ell(\delta^-)) \) and on \( \partial D_\delta(a_\ell(\delta^-)) \), showing that \( \delta \) is a point-point-edge distance.

The first containment follows from the execution of \( \text{NextEndPoint} \) at \( x^k(\delta), a^k(\delta) \) with distance \( \delta \). By the same reasoning, we also have \( x^{k+1}(\delta^-) \in \partial D_\delta(a_\ell(\delta^-)) \). As \( \delta^- \to \delta \), \( x^{k+1}(\delta^-) \to x^{k+1}(\delta) \), and \( \partial D_\delta(a_\ell(\delta^-)) \to \partial D_\delta(a_\ell(\delta^-)) \) (in the Hausdorff sense), which implies that \( x^{k+1}(\delta) \in \partial D_\delta(a_\ell(\delta^-)) \), as claimed.

Note that the distance between a point of \( A \) and an edge of \( f \) is not a critical distance (unless it coincides with a point-point-edge distance, which is ruled out anyway by our general position assumption); informally, this is because the appearance or disappearance of the small interval of intersection between the edge and the corresponding disk has no effect on \( S(\delta) \) (because there is no way to get to this interval that did not exist for \( \delta^- \) too). Note also that not all triples of two points \( p, q \) of \( A \) and an edge \( e \) of \( f \) create a point-point-edge critical event, since the bisector of \( p \) and \( q \) might not intersect \( e \).

The following lemmas lead, in combination with the decision procedure given above, to an algorithm for the optimization problem that runs in \( O((m + n)^2/3 m^{2/3} n^{1/3} \log(m + n)) \) randomized expected time. The framework of the proof is similar to that of the discrete case. Lemma 6.3 shows that, given a parameter \( L > 0 \), we can find, with high probability, an interval \( (\alpha, \beta) \) such that \( \alpha < \delta^*_F(A, f) \leq \beta \) and \( (\alpha, \beta) \) contains no more than \( L \) critical distances, in \( O(m^2 n \log(m + n) / L + (m + n) \log(m + n)) \) randomized expected time. Then, Lemma 6.4 shows that we can find \( \delta^*_F(A, f) \) within \( (\alpha, \beta) \) in \( O((m + n) L^{1/2} \log(m + n)) \) time. Choosing \( L = m^{4/3} n^{-1/3} / (m + n)^2/3 \), we obtain an algorithm that runs in \( O((m + n)^2/3 m^{2/3} n^{1/3} \log(m + n)) \) randomized expected time. The proof of Lemma 6.3 is different from that of the analogous lemma for the discrete case (Lemma 4.1), and uses a generalization of a random sampling technique by Har-Peled and Raichel (see [17]). The proof of Lemma 6.4 is similar to but more involved than the proof of the analogous lemma (Lemma 4.2) for the discrete case.

**Lemma 6.3.** Given a polygonal curve \( f \) with \( n \) edges and a set \( A \) of \( m \) points in the plane, and a parameter
$L > 0$, we can find an interval $(\alpha, \beta]$ that contains, with high probability, at most $L$ critical distances $\delta$, including $\delta^*_F(A, f)$, in $O(m^2 n \log (m + n)/L + (m + n) \log (m + n))$ randomized expected time.

**Proof.** We generate a random sample of $cx$ triples of two points of $A$ and an edge of $f$, where $x = ((\binom{m}{2} n + mn) \log (m + n)/L$, and $c > 1$ is a sufficiently large constant. We also sample $cx$ pairs of a point of $A$ and a vertex of $f$. This will generate at most $2cx$ critical values of $\delta$ (some of the triples that we sample might not contribute a critical value, as noted above).

We perform a binary search over the sampled critical values that actually arise, using the decision procedure $\Gamma$ to guide the search. This takes $O(m^2 n \log (m + n)/L + (m + n) \log (m + n))$ time (using a linear time median finding algorithm).

We claim that the interval $(\alpha, \beta]$ that this procedure generates contains, with high probability, at most $L$ (non-sampled) critical values of $\delta$, including $\delta^*_F(A, f)$. To see that, consider the set $U$ of the $L/2$ (defined) critical values that are smaller than $\delta^*_F(A, f)$ and closest to it, and denote by $u$ (resp., $v$) the number of point-vertex distances (resp., point-point-edge distances) among them; thus $u + v = L/2$. The total number of triples and pairs that potentially (but not necessarily) define a critical value is $z = ((\binom{m}{2} n + mn)/2)$, of which $(\binom{m}{2} n + mn)/2$ define point-point-edge distances, and $mn$ define point-vertex distances. The probability that none of the $2cx$ triples and pairs that we sampled generate a critical value in $U$ is at most

$$
\left(1 - \frac{u}{\binom{m}{2} n}\right)^{cx} \cdot \left(1 - \frac{v}{mn}\right)^{cx} \leq e^{-cx} \left( \frac{u}{\binom{m}{2} n} + \frac{v}{mn} \right) \leq e^{-cx} \cdot \frac{u + v}{z} = e^{-\frac{z}{2} \log (m + n)} = \frac{1}{(m + n)^{c/2}}.
$$

The same argument, with the same resulting probability bound, applies for the set $U'$ of the $L/2$ (defined) critical values that are greater than $\delta^*_F(A, f)$ and closest to it. Hence, the probability that we miss all the $L$ critical values in $U \cup U'$ is polynomially small (and can be made arbitrarily small by increasing $c$).

The following lemma shows that we can find $\delta^*_F(A, f)$ within $(\alpha, \beta]$, in $O((m + n) L^{1/2} \log (m + n))$ time. Notice the high-level similarity with the discrete counterpart in Lemma 4.2.

**Lemma 6.4.** Given a polygonal curve $f$ with $n$ edges in the plane, a set $A$ of $m$ points in the plane, and an interval $(\alpha_0, \beta_0] \subset \mathbb{R}$ that contains at most $L$ critical distances $\delta$ (including $\delta^*_F(A, f)$), we can find $\delta^*_F(A, f)$ in $O((m + n) L^{1/2} \log (m + n))$ (deterministic) time using $O(m + n)$ space.

**Proof.** For an edge $e$ of $f$ and two points $p, q \in e$, let $e[p, q]$ be the subedge of $e$ starting at $p$ (not including $p$) and ending at $q$, and let $\ell(e)$ denote the line containing $e$.

We simulate the decision procedure $\Gamma$ at the unknown value $\delta^* = \delta^*_F(A, f)$. Each step of $\Gamma$ involves a call to one of the procedures NextEndPoint and NextDisk. The execution of each of these procedures consists of a sequence of tests—the former procedure tests the current disk against a sequence of edges of $f$, for finding the first exit point from the disk, and the latter procedure tests the current point $a^{k+1}(\delta)$ against a sequence of disks centered at the points of $A$, for finding the first disk (beyond the present disk) that contains the point. Each such test generates a critical value $\delta_0$, and we check whether $\delta_0$ lies outside $(\alpha_0, \beta_0]$, in which case we know the (combinatorial nature of the) outcome of the test (in the procedure NextEndPoint, the point $a^{k+1}(\delta_0)$ itself varies continuously with $\delta_0$), and we can proceed to the next test. If $\delta_0$ lies in $(\alpha_0, \beta_0]$, we bifurcate, proceeding along two branches, one assuming that $\delta^* \leq \delta_0$ and the other assuming that $\delta^* > \delta_0$.

These bifurcations generate a tree $T$. At each node of $T$ we maintain a triple $(\tau, a^k, e^k(p, q))$, where $\tau = (\alpha, \beta]$ is a range of possible values for $\delta^*$ (a subrange of $(\alpha_0, \beta_0]$). Each such triple satisfies the
following invariant. For each $\delta \in \tau$ there exists a pair $(a^k(\delta), x^k(\delta)) \in P(\delta)$ such that $a^k(\delta) = a^k$ and $e^k[p, q]$ is the set of all points $x^k(\delta)$ for $\delta \in \tau$. In particular, $q = x^k(\beta)$ and $p$ is the limit of $x^k(\alpha^+)$ where $\alpha^+ > \alpha$ approaches $\alpha$.

The process is initialized as follows. Let $\alpha'$ be the distance between $a_1$ and $p_0$. Clearly $\delta^*_T(A, f) \geq \alpha'$. We run the decision procedure at $\alpha'$, and return $\delta^*_T(A, f) = \alpha'$ if the procedure finds a path to $(a_m, p_n)$. Otherwise we initialize the root of $T$ with the triple $((\alpha', \infty), a_1, e_1(p_0, p_0))$.

For simplicity of presentation, we represent a single cycle of the decision procedure (consisting of a call to NextEndPoint followed by a call to NextDisk) by two consecutive levels of $T$, each catering to the corresponding call. Let $v$ be a node of $T$ that represents the situation at the beginning of such a cycle. We now show how to construct the triples of the children and the grandchildren of $v$ from the triple $(\tau = (\alpha, \beta], a^k, e^k(p, q))$ of $v$.

To construct the children of $v$ we simulate NextEndPoint assuming that the current pair in $S$ is $(a^k, x^k(\delta))$ for $\delta \in \tau$ and $x^k(\delta) \in e^k(p, q)$. The idea is to compute, for $\delta = \alpha$ and for $\delta = \beta$, the edge containing the forward endpoint of the connected component of $f \cap D_\delta(a^k)$ that contains $x^k(\delta)$. If we obtain the same edge $e$ for $\delta = \alpha$ and for $\delta = \beta$, we conclude that all values in $\tau$ agree that $(a^k, e)$ is the next pair in $P$ and we continue to the next step of the procedure. Otherwise, we have detected a critical value $\delta_0$ in $\tau$ and we bifurcate, proceeding along two paths — one assuming that $\delta^* \in (\delta_0, \delta]$ and one assuming that $\delta^* \in (\delta_0, \beta]$.

We now give a more detailed description of the simulation of NextEndPoint at $v$. Let $q_0$ be the first intersection of $D_\alpha(a^k)$ with $f$ following $p = x^k(\alpha)$, and let $q_3$ be the first intersection of $D_\beta(a^k)$ with $f$ following $q = x^k(\beta)$. Let $e_j$ be the edge of $f$ equal to $e^k$. We traverse $e_j, e_{j+1}, \ldots, e_n$ in order, and for each such segment $e_\ell$, we have three possible cases.

(i) $q_0 \notin e_\ell$ and $q_3 \notin e_\ell$. In this case, the forward endpoint $x^k+1(\delta)$ of the connected component $f \cap D_\delta(a^k)$ containing $x^k(\delta)$ is not in $e_\ell$, for all $\delta \in \tau$. So we proceed to the next edge $e_{\ell+1}$.

(ii) $q_0 \in e_\ell$ and $q_3 \in e_\ell$. In this case, $e_\ell(q_0, q_3)$ is the set of all the forward endpoints $x^k+1(\delta)$ of the connected components of $f \cap D_\delta(a^k)$ containing $x^k(\delta)$, for $\delta \in \tau$. In this case $v$ has a single child $v'$, with the triple $(\tau = (\alpha, \beta], a^k, e_\ell(q_0, q_3))$.

(iii) $q_0 \in e_\ell$ and $q_3 \notin e_\ell$. In this case, we encounter a point-vertex critical distance $\delta_0 \in \tau$, between $a^k$ and $p_\ell$. That is, for each $\delta \in (\delta_0, \beta]$, the forward endpoint of the connected component $f \cap D_\delta(a^k)$ containing $x^k(\delta)$ is not in $e_\ell$ (but in a segment beyond $e_\ell$), and for each $\delta \in (\alpha, \delta_0]$, the forward endpoint of the connected component $f \cap D_\delta(a^k)$ containing $x^k(\delta)$ is in $e_\ell(q_0, p_\ell)$. We generate one child $v'$ of $v$ with the triple $((\alpha, \delta_0], a^k, e_\ell(q_0, p_\ell))$, and generate the other children of $v$ by proceeding to the following edge $e_{\ell+1}$ with the smaller range $(\delta_0, \beta]$, replacing $q_0$ with $p_\ell$ and continuing the process recursively, generating a child for each consecutive segment in which we need to bifurcate.

Next we generate the grandchildren $v''$ of each child $v'$ of $v$, which result from the simulation of the call to NextDisk. The idea is to compute, for $\delta = \alpha$ and for $\delta = \beta$, the next point $a_\ell$ of $A$ such that the disk $D_\delta(a_\ell)$ contains $x^k(\delta)$ (note that $x^k(\alpha) = p$ and $x^k(\beta) = q$). If we obtain the same point $a_\ell$ for $\delta = \alpha$ and for $\delta = \beta$, we conclude that all values in $\tau$ agree that $(a_\ell, e^k)$ is the next pair in $P$ and we continue to the next step of the procedure. Otherwise, we have detected a critical value $\delta_0$ in $\tau$ and we bifurcate, proceeding along two paths — one assuming that $\delta^* \in (\alpha, \delta_0]$ and one assuming that $\delta^* \in (\delta_0, \beta]$. We now give the details of simulating NextDisk at $v'$. For that we use the following easy observation, whose trivial proof is omitted (see Figure 7 for an illustration).

Observation 6.5. Let $a$ and $b$ be two points in the plane, let $h$ be their bisector, and let $s$ be a point on $\partial D_\delta(a)$, for some $\delta > 0$. Then $s \in D_\delta(b)$ if and only if $s$ is in the halfspace bounded by $h$ that contains $b$.

Let $v'$ be a child of $v$ corresponding to the triple $(\tau = (\alpha, \beta], a^k, e^k(p, q)]$, where $a^k$ is the point $a_\ell$. We
simulate **NextDisk** at all possible $\delta \in \tau$ by traversing $a_{i+1}, \ldots, a_m$, distinguishing between the following cases at each such point $a_\ell$.

(i) $p \notin D_\alpha(a_\ell)$ and $q \notin D_\beta(a_\ell)$. In this case, each point $x^k(\delta) \in e^k(p, q)$, for $\delta \in (\alpha, \beta]$, satisfies $x^k(\delta) \notin D_\delta(a_\ell)$. Indeed, by the way we computed the triple for $v'$, $p$ is a point on $\partial D_\alpha(a^k)$ and $q$ is a point on $\partial D_\beta(a^k)$. Thus, by Observation 6.5 $p$ and $q$ are not in the halfspace $h^+(a^k, a_l)$ bounded by $h(a^k, a_\ell)$ that contains $a_\ell$, where $h(a^k, a_\ell)$ is the bisector of $a^k$ and $a_\ell$. Thus, $x^k(\delta)$, for any $\delta \in (\alpha, \beta]$, is also not in the halfspace $h^+(a^k, a_\ell)$. Since $x^k(\delta)$ is a point on $\partial D_\delta(a^k)$, again by Observation 6.5, $x^k(\delta) \notin D_\delta(a_\ell)$. Hence, in this case, the $A$-frog cannot jump to $a_\ell$ when the person is at $x^k(\delta)$ (for each point $x^k(\delta) \in e^k(p, q)$, over $\delta \in (\alpha, \beta]$, so we proceed to the next point $a_{\ell+1}$.

(ii) $p \in D_\alpha(a_\ell)$ and $q \in D_\beta(a_\ell)$. By a similar reasoning to that of the preceding case, each point $x^k(\delta) \in e^k(p, q)$ satisfies $x^k(\delta) \in D_\delta(a_\ell)$. Hence, for each point $x^k(\delta) \in e^k(p, q)$ the $A$-frog can jump to $a_\ell$ when the person is at $x^k(\delta)$. So in this case $v'$ has only one child $v''$ that corresponds to the triple $(\tau = (\alpha, \beta), a_\ell, e^k(p, q))$.

(iii) $q \in D_\beta(a_\ell)$ and $p \notin D_\alpha(a_\ell)$. By a similar reasoning as in the previous cases, using Observation 6.5 $q$ is in the halfspace $h^+(a^k, a_l)$, and $p$ is not. Thus, there exists a point $s$ such that $s = e^k(p, q) \cap h(a^k, a_\ell)$. By the way we constructed the triple of $v'$, it follows that for each point $p' \in e^k(p, q)$ there exists a $\delta \in (\alpha, \beta]$ such that $p'$ is $x^k(\delta)$ and $x^k(\delta) \in \partial D_\delta(a^k) \cap e^k$.

By observation 6.5 we have: (a) If $x^k(\delta) \in e^k(p, s)$ then $x^k(\delta) \notin D_\delta(a_\ell)$ (so the frog cannot jump to $a_\ell$ when the person is at $x^k(\delta)$). (b) If $x^k(\delta) \in e^k(s, q)$ then $x^k(\delta) \in D_\delta(a_\ell)$ (so the frog can jump to $a_\ell$ when the person is at $x^k(\delta)$).

Let $\delta_0 = ||a^k - s|| = ||a_\ell - s||$. Clearly, $x^k(\delta) \in e^k(p, s)$ for $\alpha < \delta \leq \delta_0$ and $x^k(\delta) \in e^k(s, q)$ for $\delta_0 < \delta \leq \beta$. Further, $\delta_0$ is a point-point-edge critical value involving $a^k, a_\ell$ and $e^k$.

We generate a child $v''$ of $v'$ that corresponds to the triple $((\delta_0, \beta), a_\ell, e^k(s, q))$, and continue to generate the other children of $v'$ by proceeding to the next point (if there is one) $a_{\ell+1}$ with the updated triple $((\alpha, \delta_0), a^k, e^k(p, s))$. See Figure 8(a).

(iv) $p \in D_\alpha(a_\ell)$ and $q \notin D_\beta(a_\ell)$. Arguing similarly to the preceding case, we encounter a point-point-edge critical value $\delta_0$ involving $a^k, a_\ell$ and $e^k$, where $\delta_0$ is the distance between $s = e^k(p, q) \cap h(a^k, a_\ell)$ and $a^k$ (or $a_\ell$).

We generate a child $v''$ of $v'$ that corresponds to the triple $((\alpha, \delta_0), a_\ell, e^k(p, s))$, and continue to generate the other children of $v'$ by proceeding to the next point (if there is one) $a_{\ell+1}$ with the updated triple $((\delta_0, \beta), a^k, e^k(s, q))$. See Figure 8(b).

Note that we may reach the last point $a_m$ without generating children of $v'$ (i.e., grandchildren of $v$). In this case $\delta^*$ cannot be in $(\alpha, \beta]$, and we can abandon this branch. When we reach a node whose triple is $((\alpha, \beta], a_m, e_n(p, p_n))$ then if $\delta^* \in (\alpha, \beta]$ then $\delta^* = \beta$.

It is straightforward to show that the triple of each node $v$ that we generate satisfies the invariant men-
We stop when we reach a node that records the last step of \( S \), runs in \( O(m) \) time using \( O(m + n) \) space.

We then run a binary search over the set of \( O(m + n) \) critical values that we have accumulated at the bifurcations of \( T' \), using the decision procedure \( \Gamma \) to guide the search. This determines the leaf \( v \) of \( T' \) such that the range \( \tau \) of \( v \) contains \( \delta^* \). The path of \( T' \) leading to \( v \) is the next portion of the semi-sparse path \( S \) produced by the decision procedure (in Figure 5) at \( \delta^* \). We then repeat the whole procedure starting at \( v \). We stop when we reach a node that records the last step of \( S \), which reaches \( (a_m, p_n) \), and the final range \( (\alpha, \beta) \) of that node yields \( \delta^* = \beta \). An analysis analogous to the one in Lemma 4.2 shows that this algorithm runs in \( O((m + n)L^{1/2}\log(m + n)) \) time using \( O(m + n) \) space.

By combining Lemma 6.3 with Lemma 6.4 as noted above, we obtain the following result.

**Theorem 6.6.** Given a set \( A \) of \( m \) points and a polygonal curve \( f \) with \( n \) edges in the plane, we can compute the one-sided semi-continuous Fréchet distance \( \delta_F^*(A, f) \) with shortcuts in randomized expected time \( O((m + n)^{2/3}m^{2/3}n^{1/3}\log(m + n)) \) using \( O((m + n)^{2/3}m^{2/3}n^{1/3}) \) space.

## 7 Discussion

The algorithms obtained for the discrete Fréchet distance with shortcuts, run in time significantly better than those for the Fréchet distance without shortcuts. It is thus an interesting open question whether similar improvements can be obtained for the continuous version of the Fréchet distance with shortcuts, where
shortcuts are made only between vertices of the curves. This variant, that was considered by Driemel and Har-Peled [15], may be easier than the NP-Hard variant that was considered by Driemel et al. [11]. We hope that the techniques that we have developed for the semi-continuous problem will be useful for tackling this harder problem.

Another topic for further research is to find additional applications of some of the ideas that appear in the optimization technique for the one-sided variant.

References

[1] P. K. Agarwal, R. Ben Avraham, H. Kaplan and M. Sharir, Computing the discrete Fréchet distance in subquadratic time, Proc. 24th Annu. ACM-SIAM Sympos. Discrete Algorithms (2013), 156–167. Also in SIAM J. Comput., in press, and in arxiv:1204.5333 (2012).

[2] H. Alt and M. Godau, Computing the Fréchet distance between two polygonal curves, Internat. J. Comput. Geom. Appl. 5 (1995), 75–91.

[3] R. Ben Avraham, O. Filtser, H. Kaplan, M. J. Katz and M. Sharir, The Discrete and Semicontinuous Fréchet Distance with Shortcuts via Approximate Distance Counting and Selection, Proc. 30th ACM Sympos. Comput. Geom. (2014), to appear. arxiv:1310.5245 (2013).

[4] S. Bereg, M. Jiang, W. Wang, B. Yang and B. Zhu, Simplifying 3d polygonal chains under the discrete Frechet distance, Proc. 8th Latin American Conf. on Theoretical informatics (2008), 630–641.

[5] S. Bespamyatnikh and M. Segal, Fast algorithms for approximating distances, Algorithmica 33(2) (2002), 263–269.

[6] S. Brakatsoulas, D. Pfoser, R. Salas and C. Wenk, On map-matching vehicle tracking data, Proc. 31st Intl. Conf. Very Large Data Bases (2005), 853–864.

[7] K. Buchin, M. Buchin and J. Gudmundsson, Detecting single file movement, Proc. 16th ACM SIGSPA-TIAL Intl. Conf. Adv. GIS (2008), 288–297.

[8] K. Buchin, M. Buchin, J. Gudmundsson, M. Léffler and J. Luo, Detecting commuting patterns by clustering subtrajectories, Internat. J. Comput. Geom. Appl. 21(3) (2011), 253–282.

[9] K. Buchin, M. Buchin, W. Meulemans and W. Mulzer, Four soviets walk the dog — with an application to Alt’s conjecture, Proc. 25th Annu. ACM-SIAM Sympos. Discrete Algorithms (2014), 1399–1413.

[10] K. Buchin, M. Buchin and Y. Wang, Exact algorithms for partial curve matching via the Fréchet distance, Proc. 20th Annu. ACM-SIAM Sympos. Discrete Algorithms (2009), 645–654.

[11] M. Buchin, A. Driemel and B. Speckmann, Computing the Fréchet distance with shortcuts is NP-hard, Euro. Workshop Comput. Geom. (2013), 43–46.

[12] B. Chazelle, Cutting hyperplanes for divide-and-conquer, Discrete Comput. Geom. 9 (1993), 145–158.

[13] B. Chazelle and J. Friedman, A deterministic view of random sampling and its use in geometry, Combinatorica 10 (1990), 229–249.
[14] D. Chen, A. Driemel, L. J. Guibas, A. Nguyen and C. Wenk, Approximate map matching with respect to the Fréchet distance, *Proc. 7th Workshop on Algorithm Engineering and Experiments* (2011), 75–83.

[15] A. Driemel and S. Har-Peled, Jaywalking your dog: Computing the Fréchet distance with shortcuts, *SIAM J. Comput.* 42(5) (2013), 1830–1866.

[16] T. Eiter and H. Mannila, Computing discrete Fréchet distance, Technical Report CD-TR 94/64, Christian Doppler Laboratory for Expert Systems, TU Vienna, Austria, 1994.

[17] S. Har-Peled and B. Raichel, Net and prune: a linear time algorithm for Euclidean distance problems, *Proc. 45th ACM Sympos. Theory Comput.* (2013), 605–614.

[18] S. Har-Peled and M. Sharir, Relative $(p, \varepsilon)$-approximations in geometry, *Discrete Comput. Geom.* 45(3) (2011), 462–496.

[19] M. J. Katz and M. Sharir, An expander-based approach to geometric optimization, *SIAM J. Comput.* 26(5) (1997), 1384–1408.

[20] M. S. Kim, S. W. Kim and M. Shin, Optimization of subsequence matching under time warping in time-series databases, *Proc. ACM Sympos. Applied Comput.* (2005), 581–586.

[21] S. Kwong, Q. H. He, K. F. Man, K. S. Tang and C. W. Chau, Parallel genetic-based hybrid pattern matching algorithm for isolated word recognition, *Intl. J. Pattern Recog. Art. Intell.* 12(5) (1998), 573–594.

[22] Y. Li, P. M. Long and A. Srinivasan, Improved bounds on the sample complexity of learning, *J. Comput. Syst. Sci.* 62 (2001), 516–527.

[23] J. Matoušek, Cutting hyperplane arrangements, *Discrete Comput. Geom.* 6 (1991), 385–406.

[24] M. E. Munich and P. Perona, Continuous dynamic time warping for translation-invariant curve alignment with applications to signature verification, *Proc. 7th Intl. Conf. Comp. Vision* (1999), 108–115.

[25] M. Sharir and H. Shaul, Semialgebraic range reporting and emptiness searching with applications, *SIAM J. Comput.* 40(4) (2011), 1045–1074.

[26] C. Wenk, R. Salas and D. Pfoser, Addressing the need for map-matching speed: Localizing global curve-matching algorithms, *Proc. 18th Intl. Conf. Scientific and Statistical Database Management* (2006), 379–388.

[27] T. Wylie and B. Zhu, Protein chain pair simplification under the discrete Frechet distance, *IEEE/ACM Trans. Comput. Biology Bioinform.* (2013), 1372–1383.