Fréchet Queries in Geometric Trees*

Joachim Gudmundsson† Michiel Smid‡

September 10, 2013

Abstract

Let T be a tree that is embedded in the plane and let $\Delta > 0$ be a real number. The aim is to preprocess T into a data structure, such that, for any query polygonal path Q, we can decide if T contains a path P whose Fréchet distance $\delta_F(P, Q)$ to Q is less than Δ. For any real number $\varepsilon > 0$, we present an efficient data structure that solves an approximate version of this problem, for the case when T is c-packed and each of the edges of T and Q has length $\Omega(\Delta)$: If the data structure returns NO, then there is no such path P. If it returns YES, then $\delta_F(P, Q) \leq \sqrt{2}(1+\varepsilon)\Delta$ if Q is a line segment, and $\delta_F(P, Q) \leq 3\Delta$ otherwise.

1 Introduction

The Fréchet distance [15] is a measure of similarity between two curves P and Q that takes into account the location and ordering of the points along the curves. Let p and p' be the endpoints of P and let q and q' be the endpoints of Q. Imagine a dog walking along P and, simultaneously, a person walking along Q. The person is holding a leash that is attached to the dog. Neither the dog nor the person is allowed to walk backwards along their curve, but they can change their speeds. The Fréchet distance between P and Q is the length of the shortest leash such that the dog can walk from p to p' and the person can walk from q to q'. To define this formally, let $|xy|$ denote the Euclidean distance between two points x and y. Then the Fréchet distance $\delta_F(P, Q)$ between P and Q is defined as

$$\delta_F(P, Q) = \inf_f \max_{z \in P} |zf(z)|,$$

where f ranges over all orientation-preserving homeomorphisms $f : P \to Q$ with $f(p) = q$ and $f(p') = q'$.

*An extended abstract appears in the Proceedings of the 21st European Symposium on Algorithms, Lecture Notes in Computer Science, Volume 8125, pp. 565–576, Springer-Verlag, 2013.

†School of IT, University of Sydney, Australia. Research supported by ARC.

‡School of Computer Science, Carleton University, Ottawa, Canada. Research supported by NSERC.

Measuring the similarity between curves has been extensively studied in the last 20 years in computational geometry [1, 4, 14, 20], as well in other areas, such as data mining [16, 18], GIScience [6, 7, 21] and image processing [19].

Alt and Godau [4] showed that the Fréchet distance between two polygonal curves P and Q can be computed in $O(n^2 \log n)$ time, where n is the total number of vertices of P and Q. A lower bound of $\Omega(n \log n)$ for the decision version of the problem was given by Buchin et al. [8]. Despite extensive research, no subquadratic time algorithm is known for the problem. In 2009, Alt [2] conjectured the decision problem to be 3SUM-hard.

Recently, Agarwal et al. [1] showed how to achieve a subquadratic running time for the discrete Fréchet distance, where only the vertices of the curves are considered. Buchin et al. [9] showed how to extend their approach to the (continuous) Fréchet distance. Their algorithm takes $O(n^2\sqrt{\log n}(\log\log n)^{3/2})$ expected time.

Until recently, subquadratic time algorithms were only known for restricted cases, such as closed convex curves and κ-bounded curves [5]. In 2010, Driemel et al. [14] introduced a new class of realistic curves, the so-called c-packed curves. A polygonal path is c-packed if the total length of the edges inside any ball is bounded by c times the radius of the ball. Note that the definition generalizes naturally to geometric graphs. Driemel et al. showed that a $(1+\varepsilon)$-approximation of the Fréchet distance between two c-packed curves with a total of n vertices in $\mathbb{R}^d$ can be computed in $O(cn/\varepsilon + cn\log n)$ time. The notion of c-packedness has been argued [10, 14] to capture many realistic settings for geometric paths and graphs. Chen et al. [10] experimentally verified that maps of real world cities are ϕ-low density for a constant ϕ[1]. A geometric graph G is ϕ-low-density, if for any radius $\rho > 0$, any ball with radius ρ intersects at most ϕ edges of G that are longer than ρ. A c-packed curve is ϕ-low density for $\phi = 2c$; see [14].

In many applications, it is important to find the path in a geometric graph G that is most similar to a polygonal curve Q. Alt et al. [3] introduced this problem in 2003: Given a planar geometric graph G with n vertices and a polygonal curve Q with m vertices, the problem is to find the path P on G that has the smallest Fréchet distance to Q. They presented an algorithm that finds such a path P, with both endpoints being vertices of G, in $O(nm\log(nm)\log n)$ time using $O(nm)$ space; see also [6, 21]. The bound on the running time is close to quadratic in the worst case and, hence, unsuitable for large road maps. Chen et al. [10] considered the case when the embedding of G is ϕ-low density and the curve Q is c-packed. In $\mathbb{R}^d$, they presented a $(1+\varepsilon)$-approximation algorithm for the problem with running time $O((\phi m + cn)\log(nm)\log(n+m) + (\phi m/\varepsilon^d + cn/\varepsilon)\log(nm))$.

Very little is known about query variants of these problems. In its most general setting, the aim is to preprocess a given geometric graph G into a data structure, such that, for a polygonal path Q and a real value $\Delta > 0$ as a query, it can be decided if there exists a path P in G whose Fréchet distance to Q is less than Δ. In this setting, the path P does not have to start or end at a vertex of G.

In [13], de Berg et al. studied the case when G is a polygonal path with n vertices and Q is a single straight-line segment. For any fixed value of Δ and any parameter s with

[1] In their experiments, ϕ varied between 16 and 28.

$n \leq s \leq n^2$, they show how to build, in $O(n^2 + s\,\mathrm{polylog}(n))$ time, a data structure of size $O(s\,\mathrm{polylog}(n))$, that can be used to approximately decide if G contains a path with Fréchet distance less than Δ to any given query segment. More precisely, for any query segment Q of length more than 6Δ, the query algorithm associated with the data structure returns YES or NO in $O((n/\sqrt{s})\,\mathrm{polylog}(n))$ time.

1. If the output is YES, then there exists a path P in G such that $\delta_F(Q, P) \leq (2+3\sqrt{2})\Delta$. (In fact, their algorithm counts all such "minimal" paths.)

2. If the output is NO, then $\delta_F(Q, P) \geq \Delta$ for any path P in G.

By increasing the preprocessing time to $O(n^3 \log n)$, they show that the same result holds for the case when the threshold Δ is part of the query.

Our results: We consider the same problem as de Berg et al. [13] for the case when the graph G is a tree and a query consists of a polygonal path Q. We will write T instead of G.

Let T be a tree with n vertices that is embedded in the plane. Thus, any vertex of T is a point in $\mathbb{R}^2$ and any edge is the line segment joining its two vertices. This tree is not necessarily plane, i.e., edges of T may cross. A point x in $\mathbb{R}^2$ is said to be *on* T, if either x is a vertex of T or x is in the relative interior of some edge of T. If x and y are two points on T, then $T[x, y]$ denotes the path on T from x to y.

Let Δ be a fixed positive real number. We want to preprocess T such that queries of the following type can be answered efficiently: Given a (possibly crossing) polygonal path Q with m vertices, decide if there exist two points x and y on T, such that $\delta_F(Q, T[x, y]) < \Delta$.

Assume that the tree T is c-packed, for some constant c. Also, assume that each edge of T has length $\Omega(\Delta)$. For any constant $\varepsilon > 0$, we show that a data structure of size $O(n\,\mathrm{polylog}(n))$ can be built in $O(n\,\mathrm{polylog}(n))$ time. For any polygonal path Q with m vertices, each of whose edges has length $\Omega(\Delta)$, the query algorithm associated with the data structure returns YES or NO in $O(m\,\mathrm{polylog}(n))$ time.

1. If the output is YES and $m = 2$ (i.e., Q is a segment), then the algorithm also reports two points x and y on T such that $\delta_F(Q, T[x, y]) \leq \sqrt{2}(1 + \varepsilon)\Delta$.

2. If the output is YES and $m > 2$, then the algorithm also reports two points x and y on T such that $\delta_F(Q, T[x, y]) \leq 3\Delta$.

3. If the output is NO, then $\delta_F(Q, T[x, y]) \geq \Delta$ for any two points x and y on T.

If T is a path, then we do not need the requirement that each of its edges has length $\Omega(\Delta)$. In this case, however, we have $\delta_F(Q, T[x, y]) \leq 3(1 + \varepsilon)\Delta$ if $m > 2$.

Compared to the structure in [13], the main drawbacks are that we require the input tree T to be c-packed and all its edges to have length $\Omega(\Delta)$. However, the advantages are that (1) our structure can report a path $T[x, y]$, (2) the approximation bound is improved from $(2+3\sqrt{2})$ to $\sqrt{2}(1+\varepsilon)$ for query segments, (3) our query algorithm can handle polygonal query paths Q, (4) the preprocessing time is improved from nearly quadratic to $O(n\,\mathrm{polylog}(n))$,

(5) the input graph can be a tree and is not restricted to being a polygonal path, and (6) both the algorithm and the analysis are very simple.

The rest of this paper is organized as follows. In Section 2, we present the general approach for querying a tree with a line segment, by giving a generic query algorithm and proving its correctness. Since the running time of this algorithm can be $\Omega(n)$ for arbitrary trees, we recall c-packed trees and μ-simplifications in Section 3 and prove some of their properties. In Section 4, we use μ-simplifications to show how the generic algorithm can be implemented efficiently for querying a polygonal c-packed path with a query segment. In Section 5, we generalize the result of Section 4 to c-packed trees, all of whose edges have length $\Omega(\Delta)$. In Section 6, we use the previous results to query polygonal paths and trees with a polygonal path. Finally, in Section 7, we conclude with some directions for future work.

2 A Generic Algorithm for Query Segments

In this section, we present a generic algorithm that can be used for any tree T and any query segment Q of length more than 2Δ. We start in Section 2.1 by presenting the main technical lemmas that will be used to prove the correctness of our approach. The generic algorithm itself is presented in Section 2.2.

2.1 Technical Lemmas

Let T be a tree embedded in the plane and let $\Delta > 0$ be a real number. We assume that a query consists of a line segment Q with endpoints a and b, which we denote by $Q = [a, b]$. We also assume that $|ab| > 2\Delta$.

Throughout Section 2.1, we assume that Q is "almost" horizontal. In the generic algorithm, we will use different coordinate systems, such that this assumption holds in one of these systems for any segment Q.

Let $R(a, b)$ be the rectangle with sides of length $|ab|$ and 2Δ as indicated in Figure 1. Let D_a be the disk with center a and radius Δ, and let C_{ab} be the part of the boundary of this disk that is contained in $R(a, b)$. Define D_b and C_{ba} similarly with respect to b.

Assume that there exist two points x and y on T such that $\delta_F(Q, T[x, y]) < \Delta$. The first lemma states that we may assume that x is on C_{ab}, y is on C_{ba}, the path $T[x, y]$ is in the rectangle $R(a, b)$, and, assuming that the query segment Q is almost horizontal, when walking along $T[x, y]$ from x to y, we never travel a distance of approximately 2Δ to the left.

Lemma 1 *Let $\varepsilon > 0$ be a sufficiently small real number, let $Q = [a, b]$ be a line segment of length more than 2Δ, and assume that the angle between Q and the positive X-axis is at most ε. Assume there exist two points x and y on T, such that $\delta_F(Q, T[x, y]) < \Delta$. Then, there exist two points x' and y' on $T[x, y]$, such that the following are true:*

1. *x' and y' are on the half-circles C_{ab} and C_{ba}, respectively, and x' is on the path $T[x, y']$.*

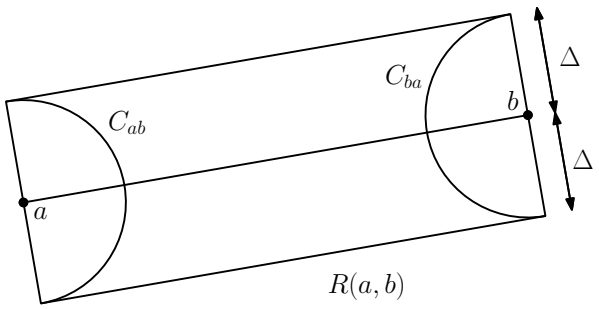

Figure 1: The rectangle $R(a,b)$ and the half-circles C_{ab} and C_{ba} corresponding to the line segment ab.

---

2. *The path $T[x',y']$ is completely contained in the region $R(a,b) \setminus (D_a \cup D_b)$.*

3. *$\delta_F(Q, T[x',y']) \leq \Delta$.*

4. *Let p and q be the first and last vertices of T on the path $T[x',y']$, respectively. For each vertex r on the path $T[p,q]$, let L_r be the vertical line through r, and let L'_r be the vertical line that is obtained by translating L_r by a distance $2(1+\varepsilon)\Delta$ to the left. Then, for each such r, the path $T[r,q]$ does not cross the line L'_r.*

Proof. Throughout this proof, we assume, without loss of generality, that the slope of Q is positive; thus, the Y-coordinate of b is at the least the Y-coordinate of a.

Let $R'(a,b)$ be the union of $R(a,b)$, D_a, and D_b. Since $\delta_F(Q, T[x,y]) < \Delta$, (i) the path $T[x,y]$ is completely contained in $R'(a,b)$, (ii) the point x is in D_a, and (iii) the point y is in D_b. Define x' to be the last point on the path $T[x,y]$ that is in D_a, and define y' to be the first point on $T[x,y]$ that is in D_b. It is clear that the first two claims in the lemma hold. The third claim follows from de Berg et al. [13, Lemma 1].

We prove the fourth claim by contradiction. Let p and q be the first and last vertices of T on the path $T[x',y']$, respectively. Assume there exists a vertex r on $T[p,q]$ such that the path $T[r,q]$ crosses the line L'_r. Then the path $T[r,q]$ contains a vertex s that is to the left of L'_r.

In the following, refer to Figure 2. Let r' and s' be the orthogonal projections of r and s onto the top side of the rectangle $R(a,b)$, respectively. Observe that s' is to the left of r'. Let a' be the intersection between the line L'_r and the top side of $R(a,b)$. Then a' is between s' and r'. Let a'' be the orthogonal projection of a' onto the bottom side of $R(a,b)$. Let b'' be the intersection between the line L_r and the bottom side of $R(a,b)$, and let b' be the orthogonal projection of b'' onto the top side of $R(a,b)$. Observe that (i) a'' and b' are between L'_r and L_r, and (ii) a' is to the left of b' and both these points are between s' and r' (on the top side of $R(a,b)$).

Finally, let α be the angle between Q and the positive X-axis, let Δ' be the horizontal distance between r and s, let h be the horizontal distance between L'_r and a'', and let h' be

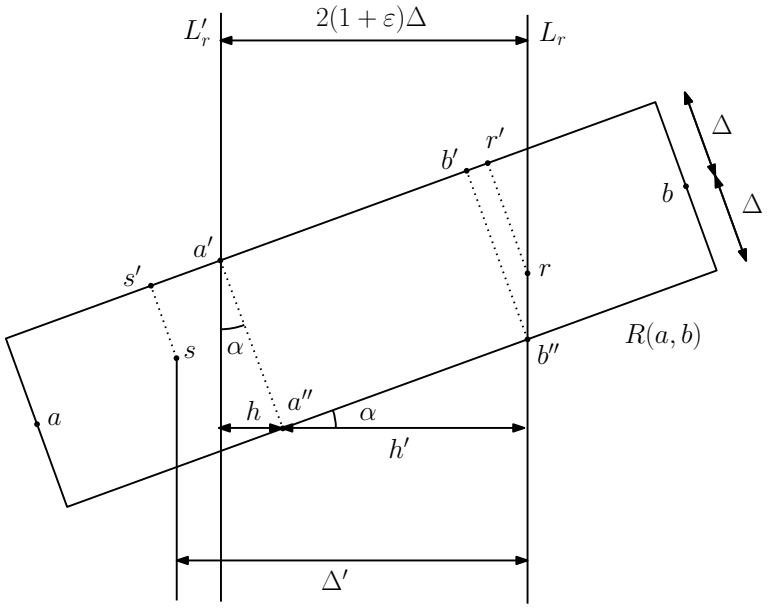

Figure 2: Proving the fourth claim in Lemma 1 by contradiction. We assume that the vertex s of the path $T[r,p]$ is to the left of the line L'_r.

---

the horizontal distance between a'' and L_r. We have

$$\begin{aligned} 2(1+\varepsilon)\Delta &= h + h' \\ &= |a'a''|\sin\alpha + |a''b''|\cos\alpha \\ &= 2\Delta\sin\alpha + |a'b'|\cos\alpha \\ &\leq 2\Delta\alpha + |a'b'| \\ &\leq 2\Delta\varepsilon + |a'b'|, \end{aligned}$$

which implies that $|a'b'| \geq 2\Delta$. Since $|r's'| > |a'b'|$, it follows that $|r's'| > 2\Delta$.

We claim that $\delta_F(Q, T[x,y]) \geq \Delta$. This will be a contradiction and, thus, complete the proof of the fourth claim in the lemma. To prove this claim, let $f : Q \to T[x,y]$ be an arbitrary orientation-preserving homeomorphism with $f(a) = x$ and $f(b) = y$. Let c and c' be the points on Q such that $f(c) = r$ and $f(c') = s$. Since s is on the path $T[r,q]$, the point c' is on the line segment $[c,b]$; refer to Figure 3. Below, we will show that

$$\max_{z \in Q} |zf(z)| > \Delta. \tag{1}$$

Since f is arbitrary, this will imply our claim that $\delta_F(Q, T[x,y]) \geq \Delta$ and, thus, complete the proof of the fourth claim in the lemma.

To prove (1), first assume that c is to the left of L'_r. Then we have

$$|cf(c)| = |cr| > 2(1+\varepsilon)\Delta > \Delta$$

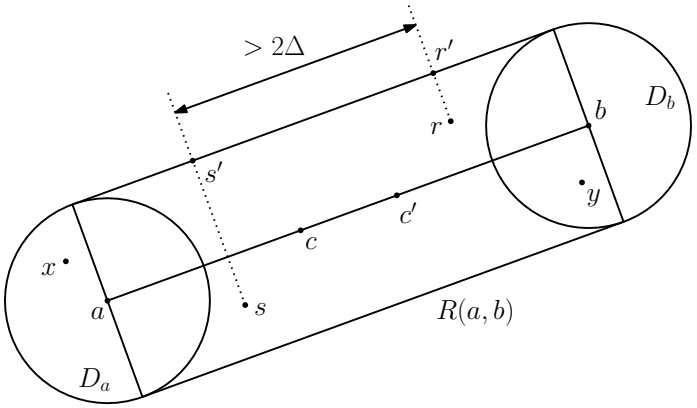

Figure 3: While traversing the path $T[x, y]$ from x to y, vertex r is visited before vertex s. While traversing the segment ab from a to b, point c is visited before point c'.

---

and, thus, (1) holds. If c is on or to the right of L'_r, then we have

$$\begin{aligned} 2\Delta &< |r's'| \\ &\leq |rs| \\ &\leq |rc| + |cs| \\ &< |rc| + |c's| \\ &= |cf(c)| + |c'f(c')|. \end{aligned}$$

It follows that $|cf(c)| > \Delta$ or $|c'f(c')| > \Delta$ and, thus, (1) holds in this case as well. ■

The next lemma considers the converse to Lemma 1 for horizontal line segments Q. It states that if there exist points x and y on T satisfying the first, second, and fourth claims in Lemma 1, then the Fréchet distance between Q and $T[x, y]$ is bounded.

Lemma 2 *Let $\Delta'' > 0$ be a real number and let $Q = [a, b]$ be a horizontal line segment of length more than 2Δ such that a is to the left of b. Assume there exist two points x and y on T, such that the following are true:*

1. *x and y are on the half-circles C_{ab} and C_{ba}, respectively.*
2. *The path $T[x, y]$ is completely contained in the region $R(a, b) \setminus (D_a \cup D_b)$.*
3. *Let p and q be the first and last vertices of T on the path $T[x, y]$, respectively. For each vertex r on the path $T[p, q]$, let L_r be the vertical line through r, and let L''_r be the vertical line that is obtained by translating L_r by a distance Δ'' to the left. Then, for each such r, the path $T[r, q]$ does not cross the line L''_r.*

Then, $\delta_F(Q, T[x, y]) \leq \sqrt{\Delta^2 + (\Delta''/2)}$.

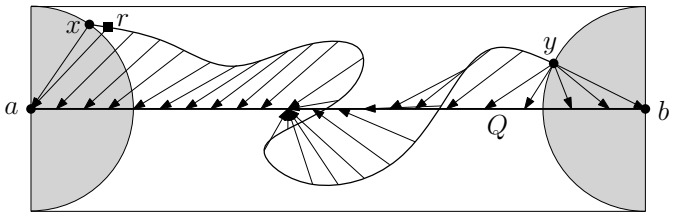

Figure 4: Illustrating the proof of Lemma 2.

Proof. First assume that $|ab| \leq \Delta''/2$. Then for each point z on Q and each point z' in $R(a,b)$, we have $|zz'| \leq \sqrt{\Delta^2 + (\Delta''/2)^2}$. Thus, for any orientation-preserving homeomorphism $f : T[x,y] \to Q$ with $f(x) = a$ and $f(y) = b$, we have

$$\max_{z \in T[x,y]} |zf(z)| \leq \sqrt{\Delta^2 + (\Delta''/2)}.$$

As a result, the lemma holds for this case. In the rest of the proof, we assume that $|ab| > \Delta''/2$. For any path P, let $\alpha(P)$ be the X-coordinate of the rightmost point of P, and let $\alpha'(P) = \alpha(P) - \Delta''/2$. Let L be the vertical line at distance $\Delta''/2$ to the right of a. Assume that y is to the right of L. Let r be the first point along $T[x,y]$ that is on or to the right of L. Imagine a dog starting at x and walking along $T[x,y]$, and a person starting at a and walking along Q, in the following way (see Figure 4):

1. The dog walks from x to r, while the person stays at a.
2. The dog walks from r to y. The person walks along Q such that for any position z of the dog, the person will be at the point on Q with X-coordinate $\alpha'(T[r,z])$.
3. Let c be the point on Q with X-coordinate $\alpha'(T[r,y])$. The dog stays at y, while the person walks from c to b.

Observe that the dog never walks backwards along $T[x,y]$ and the person never walks backwards along Q. Also, both the dog and the person make continuous walks. We claim that at any moment, the distance between the dog and the person is at most $\sqrt{\Delta^2 + (\Delta''/2)^2}$.

Consider the first part of the walk. If $r = x$, then neither the dog nor the person moves in this part. Since r is on C_{ab}, their distance is equal to Δ, which is at most $\sqrt{\Delta^2 + (\Delta''/2)^2}$. Assume that $r \neq x$. Then the dog is within a horizontal distance of $\Delta''/2$ from the person. Therefore, their distance is at most $\sqrt{\Delta^2 + (\Delta''/2)^2}$.

Consider the second part of the walk. Because of the third property in the statement of the lemma, the horizontal distance between any point z on $T[r,y]$ and the point on Q with X-coordinate $\alpha'(T[r,z])$ is at most $\Delta''/2$. As a result, the distance between the dog and the person is at most $\sqrt{\Delta^2 + (\Delta''/2)^2}$.

In the third part of the walk, as long as the person does not enter the half-circle C_{ba}, the horizontal distance to the dog will be at most $\Delta''/2$ and, thus, the distance to the dog

will be at most $\sqrt{\Delta^2 + (\Delta''/2)^2}$. After the person enters C_{ba}, the person will stay within distance $\Delta \leq \sqrt{\Delta^2 + (\Delta''/2)^2}$ from the dog.

The walks of the dog and the person do not define an orientation-preserving homeomorphism between $T[x, y]$ and Q. However, it is not difficult to see that for every $\varepsilon > 0$, there exists an orientation-preserving homeomorphism $f : T[x, y] \to Q$ with $f(x) = a$, $f(y) = b$ and

$$\max_{z \in T[x,y]} |zf(z)| \leq \sqrt{\Delta^2 + (\Delta''/2)^2} + \varepsilon.$$

Therefore, we have shown that $\delta_F(Q, T[x, y]) \leq \sqrt{\Delta^2 + (\Delta''/2)}$.

It remains to consider the case when $|ab| > \Delta''/2$ and y is on or to the left of L. In this case, we set $r = x$ and observe that the above analysis is still valid. ∎

The final lemma in this section generalizes Lemma 2 to the case when the line segment Q is almost horizontal.

Lemma 3 *Let $\varepsilon > 0$ be a sufficiently small real number, let $Q = [a, b]$ be a line segment of length more than 2Δ, and assume that the angle between Q and the positive X-axis is at most ε. Assume there exist two points x and y on T, such that the following are true:*

1. *x and y are on the half-circles C_{ab} and C_{ba}, respectively.*
2. *The path $T[x, y]$ is completely contained in the region $R(a, b) \setminus (D_a \cup D_b)$.*
3. *Let p and q be the first and last vertices of T on the path $T[x, y]$, respectively. For each vertex r on the path $T[p, q]$, let L_r be the vertical line through r, and let L'_r be the vertical line that is obtained by translating L_r by a distance $2(1 + \varepsilon)\Delta$ to the left. Then, for each such r, the path $T[r, q]$ does not cross the line L'_r.*

Then, $\delta_F(Q, T[x, y]) \leq \sqrt{2}(1 + 3\varepsilon)\Delta$.

Proof. Throughout this proof, we assume, without loss of generality, that the slope of Q is positive; thus, the Y-coordinate of b is at the least the Y-coordinate of a.

In the following, refer to Figure 5. Let r be an arbitrary vertex on the path $T[p, q]$, and let $s \neq r$ be any vertex on the path $T[r, q]$. Consider the vertical lines L_r and L_s through r and s, respectively. Let r' and s' be the orthogonal projections of r and s onto the top side of the rectangle $R(a, b)$, respectively, and assume that s' is to the left of r'. Let a'' be the intersection between L_s and the line through the bottom side of $R(a, b)$, and let a' be the orthogonal projection of a'' onto the line through the top side of $R(a, b)$. Let b' be the intersection between L_r and the line through the top side of $R(a, b)$. Let c' be the intersection between L_s and the line through the top side of $R(a, b)$. Let α be the angle between Q and the positive X-axis. Finally, let Δ' be the horizontal distance between r and s. By the

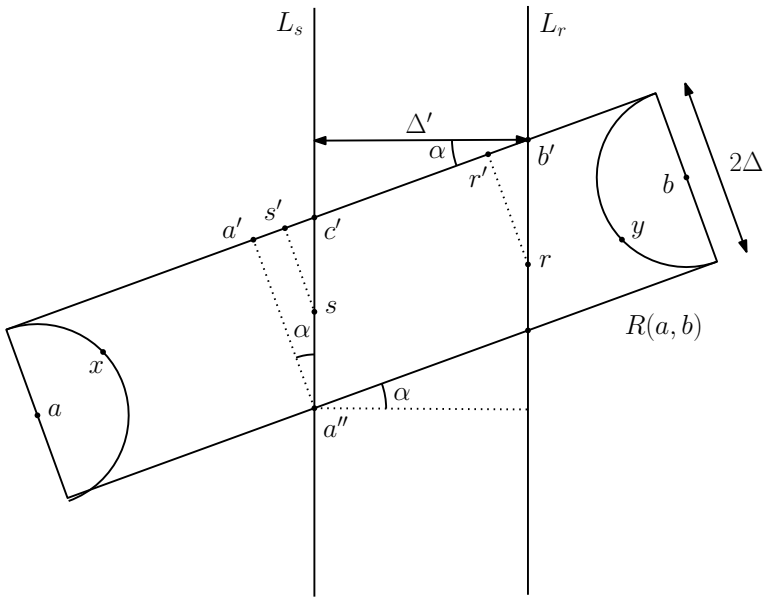

Figure 5: Illustrating the proof of Lemma 3.

---

assumption in the lemma, we have $\Delta' \leq 2(1+\varepsilon)\Delta$. We have

$$\begin{aligned}
|r's'| &\leq |a'b'| \\
&= |a'c'| + |c'b'| \\
&= |a'a''| \tan\alpha + \Delta'/\cos\alpha \\
&= 2\Delta \tan\alpha + \Delta'/\cos\alpha \\
&\leq 2\Delta \tan\varepsilon + \Delta'/\cos\varepsilon \\
&\leq 2\Delta \left(\tan\varepsilon + (1+\varepsilon)/\cos\varepsilon\right).
\end{aligned}$$

Since, for sufficiently small values of ε, $\tan\varepsilon \leq 2\varepsilon$ and $\cos\varepsilon \geq 1/(1+\varepsilon)$, it follows that

$$|r's'| \leq 2\Delta \left(2\varepsilon + (1+\varepsilon)^2\right) \leq 2\Delta\left(1+5\varepsilon\right).$$

Let $\Delta'' = 2\Delta(1+5\varepsilon)$. If we consider the line through the top side of $R(a,b)$ to be the horizontal axis, then we have shown that the conditions in Lemma 2 are satisfied. It follows that

$$\delta_F(Q, T[x,y]) \leq \sqrt{\Delta^2 + (\Delta''/2)^2} \leq \sqrt{\Delta^2 + \Delta^2(1+5\varepsilon)^2}.$$

If ε is sufficiently small, then $(1+5\varepsilon)^2 \leq 1 + 12\varepsilon$. Thus, we have shown that

$$\delta_F(Q, T[x,y]) \leq \sqrt{2}\Delta\sqrt{1+6\varepsilon} \leq \sqrt{2}\Delta(1+3\varepsilon).$$

This completes the proof. ∎

2.2 The Generic Algorithm

As before, let T be a tree embedded in the plane and let $\Delta > 0$ be a real number. Recall that, for any two points x and y on T, $T[x, y]$ denotes the path on T from x to y. We define $T(x, y)$ to be the "open" path obtained by removing the endpoints x and y from $T[x, y]$.

We choose a sufficiently small real number $\varepsilon > 0$ such that π/ε is an integer. For each integer m with $0 \leq m < \pi/\varepsilon$, consider the coordinate system, with axes X_m and Y_m, that is obtained by rotating the X-axis and Y-axis by an angle of $2m\varepsilon$.

Consider a query segment $Q = [a, b]$ of length more than 2Δ. Recall the rectangle $R(a, b)$, the disks D_a and D_b, and the half-circles C_{ab} and C_{ba} that were defined in Section 2.1.

Below, we present our generic algorithm that approximately decides if there exist two points x and y on T such that $\delta_F(Q, T[x, y]) < \Delta$. The basic idea is to choose a coordinate system in which Q is approximately horizontal. Then, using this system, we find all pairs (x, y) of points on T for which the three conditions in Lemma 3 hold.

Step 1:

- Compute the set A of intersection points between the tree T and the half-circle C_{ab}.
- Compute the set B of intersection points between the tree T and the half-circle C_{ba}.

Step 2: Compute the set $I = \{(x, y) \in A \times B : T(x, y) \cap A = \emptyset \text{ and } T(x, y) \cap B = \emptyset\}$.

Step 3: Compute the set $J = \{(x, y) \in I : T[x, y] \text{ is completely contained in } R(a, b)\}$.

Step 4:

- Let m be an index with $0 \leq m < \pi/\varepsilon$, such that the angle between Q and the positive X_m-axis is at most ε.
- For each pair (x, y) in J, do the following:
  - Let p and q be the first and last vertices of T on the path $T[x, y]$, respectively.
  - For each vertex r on $T[p, q]$, let L_r be the line through r that is parallel to the Y_m-axis, and let L'_r be the line obtained by translating L_r by a distance $2(1+\varepsilon)\Delta$ in the negative X_m-direction.
  - Decide if, for each vertex r on $T[p, q]$, the path $T[r, q]$ does not cross the line L'_r. If this is the case, then return YES together with the two points x and y, and terminate the algorithm.

If, at the end of this fourth step, the algorithm did not terminate yet, then return NO and terminate.

In the following lemma, we analyze the output of this algorithm.

Lemma 4 *Let $Q = [a, b]$ be a line segment of length more than 2Δ and consider the output of the generic algorithm on input Q.*

1. *If the output is YES, then there exist two points x and y on T such that $\delta_F(Q, T[x, y]) \leq \sqrt{2}(1 + 3\varepsilon)\Delta$.*

2. *If the output is NO, then for any two points x and y on T, $\delta_F(Q, T[x, y]) \geq \Delta$.*

Proof. Assume that the output of the algorithm is YES. Consider the two points x and y that come with this output. It follows from the algorithm that x and y satisfy the three conditions in Lemma 3. Therefore, we have $\delta_F(Q, T[x, y]) \leq \sqrt{2}(1 + 3\varepsilon)\Delta$.

To prove the second claim, assume there exist two points x and y on T such that $\delta_F(Q, T[x, y]) < \Delta$. Let x' and y' be the two points on T that satisfy the four properties in Lemma 1. Then the pair (x', y') is contained in the set J that is computed in Step 3. When the algorithm considers this pair in Step 4, it returns YES. ∎

If n denotes the number of vertices of T, then the worst-case running time of the generic algorithm will be $\Omega(n)$, because the sets A and B in Step 1 may have size $\Theta(n)$. In the following section, we recall c-packed trees and prove some of their properties. As we will see, if the tree T is c-packed, for some c that is polylogarithmic in n, and each of its edges has length $\Omega(\Delta)$, the running time of the generic algorithm will be polylogarithmic in n.

3 c-Packed Trees and μ-Simplifications

The notion of a tree being c-packed was introduced by Driemel et al. [14]. We start by recalling the definition.

Definition 1 *Let c be a positive real number and let T be a tree that is embedded in the plane. We say that T is* c-packed *if for any $\rho > 0$ and any disk D of radius ρ, the total length of $D \cap T$ is at most $c\rho$.*

In general, the arrangement induced by a tree can have a complexity that is quadratic in the number of vertices of T. The following lemma states that this cannot happen if T is c-packed.

Lemma 5 *Let T be a c-packed tree with n vertices. Then the arrangement induced by T has size $O(cn)$.*

Proof. Let e be an edge of T and let p and q be the endpoints of e. We will prove an upper bound on the number of edges that intersect e and have length at least $|e|$.

Consider the two disks D_p and D_q with radius $|e|$ that are centered at p and q, respectively. Consider any edge f of T with $|f| \geq |e|$ that intersects e. As can be seen in Figure 6, at least $(\sqrt{3}/2)|e|$ of the length of f is contained in $D_p \cup D_q$. If k is the number of such edges f, then $k(\sqrt{3}/2)|e| \leq 2c|e|$, i.e., $k \leq (4/\sqrt{3})c$.

Since T has $n-1$ edges, it follows that the arrangement induced by T has $O(cn)$ vertices. Since this arrangement is a plane graph, its number of edges and faces is $O(cn)$ as well. ∎

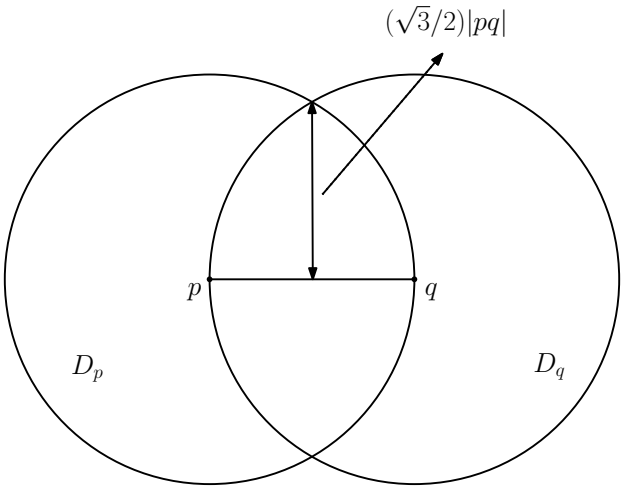

Figure 6: Any line segment that intersects pq and has length at least $|pq|$ must have at least $(\sqrt{3}/2)|pq|$ of its length in $D_p \cup D_q$.

Even if a tree T is c-packed, the number of intersection points between T and a circle C may be $\Omega(n)$. The next lemma gives an upper bound on the number of intersections for the case when the length of each edge of T is at least proportional to the radius of C.

Lemma 6 *Let T be a c-packed tree, let $\Delta > 0$ and $c' \geq 1$ be real numbers, and assume that each edge of T has length at least Δ/c'. Then, for any circle C of radius Δ, the number of intersection points between T and C is $O(cc')$.*

Proof. Let C' be the circle of radius $(1 + 1/c')\Delta$ that has the same center as C. Let e be an edge of T that intersects C. Since $|e| \geq \Delta/c'$, at least Δ/c' of the length of e is contained in C'. If k is the number of edges of T that intersect C, then $k(\Delta/c') \leq c(1 + 1/c')\Delta$, which implies that $k \leq c(c' + 1) = O(cc')$. Each of these k edges intersects C at most twice. Therefore, the number of intersection points between T and C is at most $2k$, which is $O(cc')$. ∎

In Section 4.3, we consider polygonal paths P that are c-packed, but do not have the property that each edge has length $\Omega(\Delta)$. Since Lemma 6 does not hold for such a path, we would like to simplify P, resulting in a path P', such that P' is $O(c)$-packed, each of its edges has length $\Omega(\Delta)$, and P' is close to P with respect to the Fréchet distance. Driemel et al. [14] gave a simple algorithm that computes such a path P'. Before we recall their algorithm, we formally define the notion of a simplification.

Definition 2 *Let $P = (p_1, p_2, \ldots, p_n)$ be a polygonal path in the plane and let $\mu > 0$ be a real number. A* μ-simplification *of P is a polygonal path $P' = (p_{i_1}, p_{i_2}, \ldots, p_{i_k})$ such that*

1. *$1 = i_1 < i_2 < \ldots < i_k = n$ and*

2. *each edge of P', except possibly the last one, has length at least μ.*

In order to be self-contained, we include the algorithm of [14]. The input to this algorithm is the polygonal path $P = (p_1, p_2, \ldots, p_n)$ and the real number $\mu > 0$.

$k := 1$; $i_k := 1$; $j := i_k + 1$;
while $j < n$
do while $j < n$ and $|p_{i_k} p_j| < \mu$
  do $j := j + 1$
  endwhile;
  if $j < n$
  then $k := k + 1$; $i_k := j$; $j := i_k + 1$
  endif
endwhile;
$k := k + 1$;
$i_k := n$;
return the path $P' = (p_{i_1}, p_{i_2}, \ldots, p_{i_k})$

Lemma 7 (Driemel et al. [14]) *The following are true:*

1. *In $O(n)$ time, a μ-simplification P' of P can be computed such that $\delta_F(P, P') \leq \mu$.*
2. *If the polygonal path P is c-packed, then this μ-simplification P' is $(6c)$-packed.*

Proof. A proof of the first claim can be found in Section 2.3 of [14]. The second claim is Lemma 4.3 in [14]. ∎

4 Polygonal Paths and Query Segments

In this section, we assume that the tree T is a polygonal path and write P instead of T. Thus, we assume that $P = (p_1, p_2, \ldots, p_n)$ is a polygonal path in the plane. If x and y are two points on P, then we write $x \leq_P y$ if x is on the subpath $P[p_1, y]$.

We fix a real number $\Delta > 0$ and consider queries of the following type: Given a query segment $Q = [a, b]$ of length more than 2Δ, decide if there exist two points x and y on P such that $x \leq_P y$ and $\delta_F(Q, P[x, y]) < \Delta$.

We choose positive real numbers c, c', and ε. Throughout this section, we assume that P is c-packed. We also start by assuming that each edge of P, except possibly the last one, has length at least Δ/c'. In Section 4.3, we will show how to remove the latter assumption.

4.1 Preprocessing

Arrangement $\mathcal{A}(P)$: We construct the arrangement $\mathcal{A}(P)$ induced by the path P and preprocess it for point-location queries. By Lemma 5, $\mathcal{A}(P)$ has size $O(cn)$ and, therefore, can be constructed in $O(cn \log n)$ time using a plane-sweep algorithm. After an additional $O(cn \log n)$ preprocessing time, a data structure of size $O(cn)$ can be built that supports point location queries in $O(\log n)$ time.

Circular ray shooting structures: Observe that each face of $\mathcal{A}(P)$ is a simple polygon. For each such face F, we construct the data structure $CRS(F)$ of Theorem 9 in Cheng et al. [11], which supports the following type of queries: Given any two points a and z, where z is in F and on the circle C of radius Δ centered at a, determine the first intersection between the boundary of F and the (clockwise or counter-clockwise) circular ray along C that starts at z.

If the face F has m vertices, then $CRS(F)$ has query time $O(\log m)$ (which is $O(\log n)$), size $O(m)$, and can be constructed in $O(m \log m)$ time. Since the entire arrangement $\mathcal{A}(P)$ has size $O(cn)$, the total size of all structures $CRS(F)$ is $O(cn)$ and their total construction time is $O(cn \log n)$.

Subpath rectangle intersection structure: We construct a balanced binary search tree storing the points $p_1, p_2, \ldots, p_n$ at its leaves (sorted by their indices). At each node v of this tree, we store the convex hull CH_v of all points stored in the subtree of v.

We denote the resulting data structure by $SRI(P)$. Both its size and construction time are $O(n \log n)$. Using this structure, we can answer the following type of query in $O(\log^2 n)$ time: Given any rectangle R (not necessarily axes-parallel) and any two indices i and j with $1 \leq i \leq j \leq n$, decide if the subpath $P[p_i, p_j] = (p_i, p_{i+1}, \ldots, p_j)$ of P is completely contained in R. To answer such a query, we compute $k = O(\log n)$ canonical nodes in T whose subtrees span the vertices on the subpath. For each canonical node v, we compute the four points of CH_v that are extreme in the four directions defined by the sides of R. Over all these nodes v, this gives us $4k$ points. The subpath $P[p_i, p_j]$ is completely in R if and only if all these $4k$ points are in R.

Priority search trees: In Section 2.2, we defined, for each m with $0 \leq m < \pi/\varepsilon$, the coordinate system with axes X_m and Y_m that is obtained by rotating the X-axis and Y-axis by an angle of $2m\varepsilon$.

For each index m with $0 \leq m < \pi/\varepsilon$, we do the following: For any k with $1 \leq k \leq n$, let L_{km} be the line through the point p_k that is parallel to the Y_m-axis, and let L'_{km} be the line obtained by translating L_{km} by a distance $2(1+\varepsilon)\Delta$ in the negative X_m-direction.

For each integer k with $1 \leq k \leq n$, define

$$f_m(k) = \min\{\ell : k < \ell \leq n \text{ and } p_\ell \text{ is to the left (w.r.t. the } X_m\text{-direction) of } L'_{km}\}.$$

(We define the minimum of the empty set to be ∞.) Consider the set $S_m = \{(k, f_m(k)) : 1 \leq k \leq n\}$ of n points in the plane. We construct a priority search tree $PST_m(P)$ for the set S_m

that supports range queries for three-sided rectangles that are unbounded in the negative Y-direction; see McCreight [17].

We can use $PST_m(P)$ to answer the following query in $O(\log n)$ time: Given any two indices i and j with $1 \leq i \leq j \leq n$, decide if, for each k with $i \leq k \leq j$, the subpath $P[p_k, p_j]$ does not cross the line L'_{km}. Indeed, observe that this is the case if and only if no point of S_m is in the three-sided rectangle $[i, j] \times (-\infty, j]$.

Given the set S_m, the priority search tree $PST_m(P)$ can be constructed in $O(n \log n)$ time. To compute the set S_m, we need the values $f_m(k)$, $1 \leq k \leq n$. These values are computed in the following way: For each integer k with $1 \leq k \leq n$, let p^1_{km} be the first coordinate of the point p_k in the (X_m, Y_m)-coordinate system. Define $S'_m = \{(k, p^1_{km}) : 1 \leq k \leq m\}$. Then $f_m(k)$ is equal to the first coordinate of the leftmost point of S'_m in the south-east quadrant $[k, \infty) \times (-\infty, p^1_{km} - 2(1 + \varepsilon)\Delta]$. In $O(n \log n)$ time, we construct a priority search tree for the set S'_m. We can use this structure to compute, for any k with $1 \leq k \leq n$, the value $f_m(k)$ in $O(\log n)$ time.

Thus, it takes $O((n/\varepsilon) \log n)$ time to construct all priority search trees $PST_m(P)$. Their total size is $O(n/\varepsilon)$.

Lemma 8 *The entire preprocessing algorithm takes $O((c + 1/\varepsilon)n \log n)$ time and produces a data structure of size $O(n \log n + (c + 1/\varepsilon)n)$.*

4.2 The Query Algorithm

Let $Q = [a, b]$ be a query line segment of length more than 2Δ. Recall that we assume that the polygonal path P is c-packed and each edge of P, except possibly the last one, has length at least Δ/c'.

We show how the data structures from Section 4.1 can be used to implement the four steps of the generic algorithm of Section 2.2.

Step 1: Consider the half-circle C_{ab} and let z be one of its endpoints. We first find the face F in the arrangement $\mathcal{A}(P)$ that contains z. Then we use $CRS(F)$ to find the first intersection between the boundary of F and the circular ray along C_{ab} that starts at z. This gives us the first intersection, say x, between P and C_{ab}. We then set z to x and repeat this procedure. In this way, we obtain the set A of intersection points between P and C_{ab}. In a similar manner, we obtain the set B of intersection points between P and the half-circle C_{ba}.

By Lemma 6, both A and B have size $O(cc')$. Therefore, the time for this part of the query algorithm is $O((|A| + |B|) \log n)$, which is $O(cc' \log n)$.

Step 2: We sort the points of $A \cup B$ in the order in which they occur along the path P. By scanning the sorted order, we obtain the set

$$I = \{(x, y) \in A \times B : x \leq_P y, P(x, y) \cap A = \emptyset \text{ and } P(x, y) \cap B = \emptyset\}.$$

Since $|A| + |B| = O(cc')$, this part of the query algorithm takes $O(cc' \log(cc'))$ time.

Step 3: For each element (x, y) in I, we use the data structure $SRI(P)$ to decide if the subpath $P[x, y]$ is completely contained in the rectangle $R(a, b)$. If this is the case, then we add the pair (x, y) to an initially empty set.

At the end of this step, we have computed the set

$$J = \{(x, y) \in I : P[x, y] \text{ is completely contained in } R(a, b)\}.$$

Since the size of I is $O(cc')$, this part of the query algorithm takes $O(cc' \log^2 n)$ time.

Step 4: Let m be an index with $0 \leq m < \pi/\varepsilon$, such that the angle between Q and the positive X_m-axis is at most ε. For each pair (x, y) in the set J, we do the following: If x and y are on the same edge of P, then return YES together with the two points x and y, and terminate the algorithm. Otherwise, let p_i and p_j be the first and last vertices of P on the path $P[x, y]$, respectively. Observe that $i \leq j$. Use the priority search tree $PST_m(P)$ to decide if, for each k with $i \leq k \leq j$, the subpath $P[p_k, p_j]$ does not cross the line L'_{km}. If this is the case, then return YES together with the two points x and y, and terminate the algorithm.

If, at the end of this fourth step, the algorithm did not terminate yet, then return NO and terminate.

Since the size of J is $O(cc')$, this part of the query algorithm takes $O(cc' \log n)$ time.

This concludes the description of the query algorithm. It is clear that these four steps correctly implement the generic algorithm of Section 2.2.

Lemma 9 *The query algorithm takes $O(cc' \log^2 n)$ time and correctly implements the generic algorithm of Section 2.2. Thus, the two claims in Lemma 4 hold for the output of this algorithm.*

Note that this result assumes that each edge of P, except possibly the last one, has length at least Δ/c'. In the next section, we remove this assumption.

4.3 General c-Packed Paths

Let $P = (p_1, p_2, \ldots, p_n)$ be a polygonal path and assume that P is c-packed for some real number $c > 0$. As before, we choose real numbers Δ, ε, and c'.

Let $\mu = \Delta/c'$. In $O(n)$ time, we compute a μ-simplification P' of P; see Lemma 7. Then we run the preprocessing algorithm of Section 4.1 on P'. For any given query segment $Q = [a, b]$ of length more than 2Δ, we run the query algorithm of Section 4.2 on the data structure for P'.

Assume the output of the query algorithm is YES. By Lemma 4, there exist two points x' and y' on P' such that $x' \leq_{P'} y'$ and $\delta_F(Q, P'[x', y']) \leq \sqrt{2}(1 + 3\varepsilon)\Delta$. Since, by Lemma 7, $\delta_F(P, P') \leq \mu = \Delta/c'$, it follows that there exist two points x and y on P such that $x \leq_P y$ and

$$\delta_F(Q, P[x, y]) \leq \sqrt{2}(1 + 3\varepsilon)\Delta + \Delta/c' = \sqrt{2}\Delta(1 + 3\varepsilon + 1/(c'\sqrt{2})). \tag{2}$$

On the other hand, if the output of the query algorithm is NO, then we have, by Lemma 4, $\delta_F(Q, P'[x', y']) \geq \Delta$ for any two points x' and y' on P' with $x' \leq_{P'} y'$. Therefore, we have

$$\delta_F(Q, P[x, y]) \geq \Delta - \mu = (1 - 1/c')\Delta \tag{3}$$

for any two points x and y on P with $x \leq_P y$.

By taking $c' = 1/\varepsilon$ and defining $\Delta_0 = (1 - 1/c')\Delta$, the right-hand side in (2) becomes

$$\sqrt{2}\Delta_0(1 + O(\varepsilon)),$$

whereas the right-hand side in (3) becomes Δ_0. Thus, by replacing ε in the entire construction by ε/c'', for some sufficiently large constant c'', we have proved the following result[2].

Theorem 1 *Let P be a polygonal path in the plane with n vertices, let c and Δ be positive real numbers, and assume that P is c-packed. For any $\varepsilon > 0$, we can construct a data structure of size $O(n \log n + (c + 1/\varepsilon)n)$ in $O((c + 1/\varepsilon)n \log n)$ time. Given any segment Q of length more than 2Δ, the query algorithm corresponding to this data structure takes $O((c/\varepsilon) \log^2 n)$ time and outputs either YES or NO.*

1. *If the output is YES, then there exist two points x and y on P with $x \leq_P y$ such that $\delta_F(Q, P[x, y]) \leq \sqrt{2}(1 + \varepsilon)\Delta$.*
2. *If the output is NO, then $\delta_F(Q, P[x, y]) \geq \Delta$ for any two points x and y on P with $x \leq_P y$.*

5 Trees and Query Segments

Let T be a tree with n vertices that is embedded in the plane. We fix a real number $\Delta > 0$ and consider queries of the following type: Given a query segment $Q = [a, b]$ of length more than 2Δ, decide if there exist two points x and y on T such that $\delta_F(Q, T[x, y]) < \Delta$.

We choose positive real numbers c, c', and ε, and assume that T is c-packed and each edge of T has length at least Δ/c'.

5.1 The Path Decomposition

In this section, we recall a technique due to Cole and Vishkin [12] that decomposes the tree T into a collection of paths such that any path in T overlaps $O(\log n)$ paths in the decomposition.

Fix one vertex of T and call it the root. For any vertex v of T, the *subtree* of v is the set of all vertices u of T such that v is on the path between u and the root. Let $size(v)$ denote the number of vertices in the subtree of v, and let $\ell(v) = \lfloor \log(size(v)) \rfloor$. Thus, $\ell(v)$ is an integer in $\{0, 1, \ldots, \lfloor \log n \rfloor\}$.

[2]In the theorem, we rename Δ_0 as Δ.

For a given integer ℓ, consider a maximal subtree of T consisting of vertices v with $\ell(v) = \ell$. This subtree is in fact a path which is contained in some root-to-leaf path. Thus, the values $\ell(v)$ induce a partition of the vertex set of V into a collection of paths. Note that no two paths in this partition share a vertex. For any path in the partition, let v be the vertex on this path that is closest to the root (with respect to distances along T). If v is not the root, then we add the parent of v to the path. The resulting collection of paths is called the *path decomposition* $PD(T)$ of the tree T. Any two paths in $PD(T)$ are edge-disjoint and can have at most one vertex in common.

For any two points x and y on T, the path $T[x, y]$ overlaps $O(\log n)$ paths in $PD(T)$. More precisely, given x and y, in $O(\log n)$ time, a sequence $v_1, \ldots, v_k$ can be computed, such that

1. $k = O(\log n)$,
2. $v_1 = x$ and $v_k = y$,
3. for each i with $2 \leq i \leq k-1$, v_i is an endpoint of some path in $PD(T)$ (and, thus, a vertex of T),
4. for each i with $1 \leq i < k$, the path $T[v_i, v_{i+1}]$ is contained in some path in $PD(T)$,
5. the path $T[x, y]$ in T between x and y is equal to the concatenation of the paths $T[v_1, v_2], T[v_2, v_3], \ldots, T[v_{k-1}, v_k]$.

Using this, the following type of query can be answered in $O(\log n)$ time: Given three points x, y, and z on T, together with the edges that contains them, decide if z is on the path $T[x, y]$.

5.2 Preprocessing

We construct the arrangement $\mathcal{A}(T)$ induced by the tree T and preprocess it for point-location queries. For each face F of $\mathcal{A}(T)$, we construct the data structure $CRS(F)$ for circular ray shooting queries, as in Section 4.1. Using Lemma 5, this takes $O(cn \log n)$ time and uses $O(cn)$ space.

Next, we compute the path decomposition $PD(T)$ of T, which can be done in $O(n)$ time. For each path P in $PD(T)$, we construct the data structure $SRI(P)$ of Section 4.1. Also, for each path P in $PD(T)$ and for each m with $0 \leq m < \pi/\varepsilon$, we construct two data structures $PST^{\rightarrow}_M(P)$ and $PST^{\leftarrow}_M(P)$, as in Section 4.1, one structure for each direction in which the path P can be traversed. This part of the preprocessing takes $O((n/\varepsilon) \log n)$ time and results in a data structure of size $O(n/\varepsilon + n \log n)$.

Additionally, we do the following:

Leftmost and rightmost structures: Consider the (X_m, Y_m)-coordinate systems, $0 \leq m < \pi/\varepsilon$. For each such index m and each path P in $PD(T)$, we construct a balanced binary search tree $LR_m(P)$ storing the points of P at its leaves, in the order in which they appear along P. At each node v of this tree, we store the point in the subtree of v that is extreme in the positive X_m-direction and the point in the subtree of v that is extreme in the negative X_m-direction.

The total time to compute all trees $LR_m(P)$ is $O((n/\varepsilon)\log n)$ and their total size is $O(n/\varepsilon)$.

For any given path P in $PD(T)$, any index m, and any two points x and y on P, we can use $LR_m(P)$ to compute, in $O(\log n)$ time, the point on $P[x, y]$ that is extreme in the positive X_m-direction and the point on $P[x, y]$ that is extreme in the negative X_m-direction.

The entire preprocessing algorithm takes $O((c + 1/\varepsilon)n \log n)$ time. The resulting data structure has size $O(n \log n + (c + 1/\varepsilon)n)$.

5.3 The Query Algorithm

Consider a query segment $Q = [a, b]$ of length more than 2Δ. We show how the data structures can be used to implement the four steps of the generic algorithm of Section 2.2.

Step 1: In exactly the same way as in Section 4.2, we compute, in $O(cc' \log n)$ time, the sets A and B of intersection points between T and the half-circles C_{ab} and C_{ba}, respectively. Observe that both A and B have size $O(cc')$.

Step 2: For each x in A and each y in B, we do the following: Consider all points $z \in (A \cup B) \setminus \{x, y\}$ and, for each such point, decide if it is on the path $T[x, y]$. If this is not the case for all such z, then we add the pair (x, y) to an initially empty set.

At the end of this step, we have computed the set

$$I = \{(x, y) \in A \times B : T(x, y) \cap A = \emptyset \text{ and } T(x, y) \cap B = \emptyset\}.$$

The total time for this step is $O((cc')^3 \log n)$ and the size of the set I is $O((cc')^2)$.

Step 3: For each (x, y) in I, we do the following: Use the path decomposition $PD(T)$ to compute the sequence of $O(\log n)$ paths in $PD(T)$ that overlap $T[x, y]$. For each such path P, use the data structure $SRI(P)$ to decide if the maximal subpath of P that is on $T[x, y]$ is completely contained in the rectangle $R(a, b)$. If this is the case for each such P, then we add the pair (x, y) to an initially empty set.

At the end of this step, we have computed the set

$$J = \{(x, y) \in I : T[x, y] \text{ is completely contained in } R(a, b)\}.$$

To bound the running time of this step, observe that, for each pair (x, y) in I, we spend $O(\log^2 n)$ time for each of the $O(\log n)$ paths P. Thus, since I has size $O((cc')^2)$, the total time for this step is $O((cc')^2 \log^3 n)$.

Step 4: Let m be an index with $0 \leq m < \pi/\varepsilon$, such that the angle between Q and the positive X_m-axis is at most ε. Consider any pair (x, y) in the set J. Let p and q be the first and last vertices on the path $T[x, y]$, respectively. For each vertex r on $T[p, q]$, let L_r be the line through r that is parallel to the Y_m-axis, and let L'_r be the line obtained by translating L_r by a distance of $2(1 + \varepsilon)\Delta$ in the negative X_m-direction. We have to decide if for each such vertex r, the path $T[r, q]$ does not cross the line L'_r.

We compute the sequence $v_1, \ldots, v_k$ as in Section 5.1. For each integer i with $1 \leq i < k$, let P_i be the path in the path decomposition $PD(T)$ that contains $T[v_i, v_{i+1}]$.

Assume there are two vertices r and s on $T[p, q]$, such that s is on $T[r, q]$ and s is to the left of L'_r (with respect to the X_m-direction). Let i and j be the indices such that r is on the path P_i and s is on the path P_j. Observe that $i \leq j$. There are two possibilities:

1. If $i = j$, then we use one of the two priority search trees $PST^{\rightarrow}_M(P_i)$ and $PST^{\leftarrow}_M(P_i)$ to find such vertices r and s in $O(\log n)$ time.

   Thus, by considering each of the $k = O(\log n)$ values of i, we can handle this case in $O(\log^2 n)$ time.

2. If $i < j$, then we may assume that r is the vertex on $P_i \cap T[p, q]$ that is extreme in the positive X_m-direction, whereas s is the vertex on the concatenation of the paths $P_{i+1} \cap T[p, q], \ldots, P_{k-1} \cap T[p, q]$ that is extreme in the negative X_m-direction. Thus, we handle this case in the following way:

   For each integer i with $1 \leq i < k$, use the data structure $LR_m(P_i)$ to find the two points r_i and s_i on $P_i \cap T[p, q]$ that are extreme in the positive and negative X_m-direction, respectively. Then for $i = k - 2, k - 3, \ldots, 1$, find the point s'_i on the concatenation of the paths $P_{i+1} \cap T[p, q], \ldots, P_{k-1} \cap T[p, q]$ that is extreme in the negative X_m-direction. Finally, for each i with $1 \leq i < k$, check if the point s'_i is to the left of the line L'_{r_i}.

   The total time to handle this case is $O(\log^2 n)$.

Thus, for each of the $O((cc')^2)$ pairs (x, y) in the set J, we spend $O(\log^2 n)$ in this step of the query algorithm. It follows that the total time for Step 4 is $O((cc')^2 \log^2 n)$.

This concludes the description of the query algorithm. Observe that it correctly implements the generic algorithm of Section 2.2. Therefore, we have proved the following result.

Theorem 2 *Let T be a tree with n vertices that is embedded in the plane, and let c, c', and Δ be positive real numbers. Assume that T is c-packed and each of its edges has length at least Δ/c'. For any $\varepsilon > 0$, we can construct a data structure of size $O(n \log n + (c+1/\varepsilon)n)$ in $O((c+1/\varepsilon)n \log n)$ time. Given any segment Q of length more than 2Δ, the query algorithm corresponding to this data structure takes $O((cc')^2 \log^3 n + (cc')^3 \log n)$ time and outputs either YES or NO.*

1. *If the output is YES, then there exist two points x and y on T such that $\delta_F(Q, T[x, y]) \leq \sqrt{2}(1 + \varepsilon)\Delta$.*

2. *If the output is NO, then* $\delta_F(Q, T[x, y]) \geq \Delta$ *for any two points* x *and* y *on* T.

Unfortunately, the simplification technique of Section 3 cannot be used to remove the assumption that each edge of T has length at least Δ/c': The number of paths in the path decomposition $PD(T)$ of the tree T can be $\Omega(n)$. Therefore, if we apply the simplification technique to each such path, as we did in Section 4.3, we may get $\Omega(n)$ simplified paths, each of which contains one edge of length less than $\mu = \Delta/c'$. As a result, the sets A and B that are computed in Step 1 of the query algorithm may have linear size.

6 Querying with a Polygonal Path

Until now, we have considered querying polygonal paths or trees with a line segment. In this section, we generalize our results to queries Q consisting of a polygonal path. Unfortunately, the approximation factor increases from $\sqrt{2}(1+\varepsilon)$ to 3, and we need the requirement that each edge of Q has length more than 4Δ.

Let T be a tree embedded in the plane and let $\Delta > 0$ be a real number. We consider polygonal query paths $Q = (q_1, q_2, \ldots, q_m)$, all of whose edges have length more than 4Δ. As before, we fix a sufficiently small real number $\varepsilon > 0$. The following two lemmas (which only need the requirement that each edge of Q has length more than 2Δ) generalize Lemmas 1 and 3.

Lemma 10 *Let* $Q = (q_1, q_2, \ldots, q_m)$ *be a polygonal path, all of whose edges have length more than* 2Δ. *Assume there exist two points* x *and* y *on* T, *such that* $\delta_F(Q, T[x, y]) < \Delta$. *Then, there exist points* $x'_1, y'_1, \ldots, x'_{m-1}, y'_{m-1}$ *on* $T[x, y]$, *such that the following are true:*

1. *By traversing the path* $T[x, y]$, *we visit the points* $x'_1, y'_1, \ldots, x'_{m-1}, y'_{m-1}$ *in this order.*
2. *For each integer* i *with* $1 \leq i < m$,
   (a) x'_i *and* y'_i *are on the half-circles* $C_{q_i q_{i+1}}$ *and* $C_{q_{i+1} q_i}$, *respectively,*
   (b) *the path* $T[x'_i, y'_i]$ *is completely contained in the region* $R(q_i, q_{i+1}) \setminus (D_{q_i} \cup D_{q_{i+1}})$.
3. *For each integer* i *with* $1 \leq i < m-1$, *the path* $T[y'_i, x'_{i+1}]$ *is completely contained in the disk with center* q_{i+1} *and radius* 3Δ.
4. $\delta_F(Q, T[x'_1, y'_{m-1}]) \leq \Delta$.
5. *For each integer* i *with* $1 \leq i < m$, *let* m_i *be an index such that the angle between the line segment* $[q_i, q_{i+1}]$ *and the positive* X_{m_i}*-axis is at most* ε. *Let* p_i *and* q_i *be the first and last vertices of* T *on the path* $T[x'_i, y'_i]$, *respectively. For each vertex* r *on the path* $T[p_i, q_i]$, *let* L_r *be the line through* r *that is parallel to the* Y_{m_i}*-axis, and let* L'_r *be the line that is obtained by translating* L_r *by a distance* $2(1+\varepsilon)\Delta$ *in the negative* X_{m_i}*-direction. Then, for each such* r, *the path* $T[r, q_i]$ *does not cross the line* L'_r.

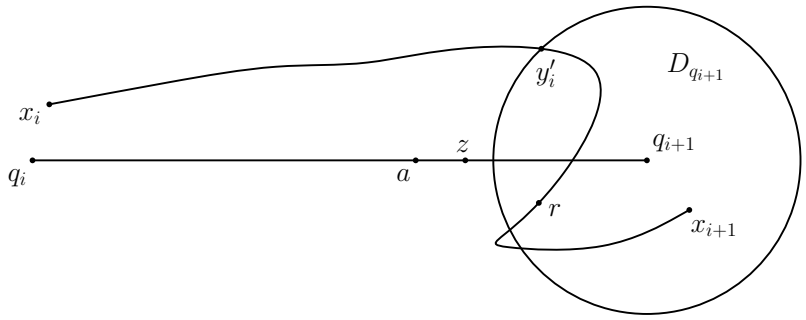

Figure 7: Illustrating the proof of the third claim in Lemma 10. The points a and y'_i are matched by the orientation-preserving homeomorphism $f : Q \to T[x, y]$. The points z and r are matched by f.

Proof. Since $\delta_F(Q, T[x, y]) < \Delta$, there exists an orientation-preserving homeomorphism $f : Q \to T[x, y]$ such that $f(q_1) = x$, $f(q_m) = y$ and $\max_{z \in Q} |zf(z)| < \Delta$. For each integer i with $1 \leq i \leq m$, define $x_i = f(q_i)$.

Let i be any index such that $1 \leq i < m$. It is obvious that $\delta_F([q_i, q_{i+1}], T[x_i, x_{i+1}]) < \Delta$. Therefore, we can apply Lemma 1 to the line segment $[q_i, q_{i+1}]$ and the two points x_i and x_{i+1} on T. We obtain two points x'_i and y'_i on the path $T[x_i, x_{i+1}]$ such that the first, second, and fifth claims hold. It is not difficult to see that the fourth claim also holds.

It remains to prove the third claim. Let i be any index such that $1 \leq i < m - 1$. Recall from the proof of Lemma 1 that y'_i is the first point on the path $T[x_i, x_{i+1}]$ that is in the disk $D_{q_{i+1}}$, whereas x'_{i+1} is the last point on $T[x_{i+1}, x_{i+2}]$ that is in $D_{q_{i+1}}$. We have to show that $|q_{i+1}r| \leq 3\Delta$ for each point r on the path $T[y'_i, x'_{i+1}]$. We will prove this for the case when r is on $T[y'_i, x_{i+1}]$; the other case can be proved in a symmetric way.

In the following, refer to Figure 7. Consider an arbitrary point r on $T[y'_i, x_{i+1}]$. Let a be the point on $[q_i, q_{i+1}]$ such that $f(a) = y'_i$, and let z be the point on $[a, q_{i+1}]$ such that $f(z) = r$. We have

$$\begin{aligned} |q_{i+1}r| &\leq |q_{i+1}z| + |zr| \\ &\leq |q_{i+1}a| + |zr| \\ &\leq |q_{i+1}y'_i| + |y'_i a| + |zr| \\ &= \Delta + |af(a)| + |zf(z)| \\ &\leq 3\Delta. \end{aligned}$$

This completes the proof of the lemma. ∎

Lemma 11 *Let $Q = (q_1, q_2, \ldots, q_m)$ be a polygonal path, all of whose edges have length more than 2Δ. Assume there exist points $x_1, y_1, \ldots, x_{m-1}, y_{m-1}$ on T, such that the following are true:*

1. *By traversing the path $T[x_1, y_{m-1}]$, we visit the points $x_1, y_1, \ldots, x_{m-1}, y_{m-1}$ in this order.*

2. *For each integer i with $1 \leq i < m$,*

   (a) *x_i and y_i are on the half-circles $C_{q_i q_{i+1}}$ and $C_{q_{i+1} q_i}$, respectively,*

   (b) *the path $T[x_i, y_i]$ is completely contained in the region $R(q_i, q_{i+1}) \setminus (D_{q_i} \cup D_{q_{i+1}})$.*

3. *For each integer i with $1 \leq i < m-1$, the path $T[y_i, x_{i+1}]$ is completely contained in the disk with center q_{i+1} and radius 3Δ.*

4. *For each integer i with $1 \leq i < m$, let m_i be an index such that the angle between the line segment $[q_i, q_{i+1}]$ and the positive X_{m_i}-axis is at most ε. Let p_i and q_i be the first and last vertices of T on the path $T[x_i, y_i]$, respectively. For each vertex r on the path $T[p_i, q_i]$, let L_r be the line through r that is parallel to the Y_{m_i}-axis, and let L'_r be the line that is obtained by translating L_r by a distance $2(1+\varepsilon)\Delta$ in the negative X_{m_i}-direction. Then, for each such r, the path $T[r, q_i]$ does not cross the line L'_r.*

Then, $\delta_F(Q, T[x_1, y_{m-1}]) \leq 3\Delta$.

Proof. It follows from Lemma 3 that

$$\delta_F([q_i, q_{i+1}], T[x_i, y_i]) \leq \sqrt{2}(1+3\varepsilon)\Delta \leq 3\Delta,$$

for each i with $1 \leq i < m$, assuming that ε is sufficiently small. By the third requirement in the lemma, we have $|q_{i+1} r| \leq 3\Delta$ for each i with $1 \leq i < m-1$ and each point r on the path $T[y_i, x_{i+1}]$. From these two claims, the lemma follows. ∎

Our algorithm will run the query algorithms of the previous sections separately on each edge of the query path Q. Afterwards, we check if the partial paths in the tree T obtained for the edges of Q can be combined into one global path that is close to the entire path Q with respect to the Fréchet distance. The following lemma will imply that this combining step is possible if such a global path exists. We remark that this is where we need the assumption that each edge of Q has length more than 4Δ.

Lemma 12 *Let $P = (p_1, p_2, \ldots, p_n)$ and $Q = (q_1, q_2, \ldots, q_m)$ be polygonal paths and assume that each edge of Q has length more than 4Δ. Let i be any index with $1 \leq i < m-1$ and assume that there exists a point y on P, such that y is on the half-circle $C_{q_{i+1} q_i}$. Then there exists at most one pair (x', y') of points on P such that*

1. *$y \leq_P x' \leq_P y'$,*

2. *x' and y' are on the half-circles $C_{q_{i+1} q_{i+2}}$ and $C_{q_{i+2} q_{i+1}}$, respectively,*

3. *the path $P[x', y']$ is completely contained in the region $R(q_{i+1}, q_{i+2}) \setminus (D_{q_{i+1}} \cup D_{q_{i+2}})$,*

4. *the path $P[y, x']$ is completely contained in the disk with center q_{i+1} and radius 3Δ.*

Proof. Assume that there exist two such pairs (x', y') and (x'', y'') of points on P. It follows from the second and third items that the paths $P[x', y']$ and $P[x'', y'']$ do not overlap. Thus, we may assume without loss of generality that $y' \leq_P x''$. We have

$$4\Delta < |q_{i+1}q_{i+2}| \leq |q_{i+1}y'| + |y'q_{i+2}| = |q_{i+1}y'| + \Delta,$$

and, therefore, $|q_{i+1}y'| > 3\Delta$. Since y' is on the path $P[y, x'']$, it follows that the path $P[y, x'']$ is not completely contained in the disk with center q_{i+1} and radius 3Δ. This is a contradiction. ∎

6.1 Querying a Polygonal Path with a Polygonal Path

Assume that the tree T is a polygonal path. As before, we write P instead of T. Thus, we assume that $P = (p_1, p_2, \ldots, p_n)$ is a polygonal path in the plane.

We choose positive real numbers Δ, c, c', and ε, and assume that P is c-packed. As before, we start by assuming that each edge of P, except possibly the last one, has length at least Δ/c'.

Preprocessing: We run the preprocessing algorithm of Section 4.1. Additionally, we do the following:

Subpath furthest point structure: We construct a balanced binary search tree storing the points $p_1, \ldots, p_n$ at its leaves, sorted by their indices. At each node v of this tree, we store the furthest-point Voronoi diagram FVD_v of all points stored in the subtree of v.

We denote the resulting data structure by $SFP(P)$. This structure has size $O(n \log n)$ and can be constructed in $O(n \log^2 n)$ time. For any disk D in the plane (given by its center and radius) and any two indices i and j with $1 \leq i \leq j \leq n$, we can use $SFP(P)$ to decide if the subpath $P[p_i, p_j]$ is completely contained in D. The time to answer such a query is $O(\log^2 n)$.

The entire preprocessing algorithm takes $O(n \log^2 n + (c + 1/\varepsilon)n \log n)$ time and results in a data structure of size $O(n \log n + (c + 1/\varepsilon)n)$.

Answering a query: Consider a query path $Q = (q_1, q_2, \ldots, q_m)$ all of whose edges have length more than 4Δ.

For each integer i with $1 \leq i < m$, we run Steps 1, 2, and 3 of the query algorithm of Section 4.2 with the segment $[q_i, q_{i+1}]$. This gives the set J_i of all pairs (x, y) of points such that

- $x \leq_P y$,
- x is an intersection point between P and the half-circle $C_{q_i q_{i+1}}$,
- y is an intersection point between P and the half-circle $C_{q_{i+1} q_i}$,

- the path $P[x, y]$ is completely contained in the region $R(q_i, q_{i+1}) \setminus (D_{q_i} \cup D_{q_{i+1}})$.

Next, we change Step 4 of the algorithm in Section 4.2, again with the segment $[q_i, q_{i+1}]$, as follows. We consider all pairs (x, y) in J_i. For each such pair, if the test in Step 4 is positive, then we insert this pair into an initially empty set K_i.

At the end of Step 4, K_i is the set of all pairs (x, y) satisfying the three properties in Lemma 3, with respect to the line segment $[q_i, q_{i+1}]$.

After this fourth step, we proceed as follows. Initialize $K'_1 = K_1$. For $i = 1, 2, \ldots, m-2$, do the following: At this moment, we have a set K'_i. Initialize $K'_{i+1} = \emptyset$. Consider each pair in K'_i. For each such pair (x, y), decide if there exists a pair (x', y') in K_{i+1} such that (i) $y \leq_P x'$ and (ii) the path $P[y, x']$ is completely contained in the disk with center q_{i+1} and radius 3Δ. If this is the case, then we insert the pair (x, y') into the set K'_{i+1}. At the end of this algorithm, we have obtained a set K'_{m-1}. If this set is non-empty, then we return YES; otherwise, we return NO.

The set K'_1 (which is equal to K_1) has size $O(cc')$. It then follows from Lemma 12 that each of the sets $K'_1, K'_2, \ldots, K'_{m-1}$ has size $O(cc')$ as well. This implies that the total time to answer a query is $O((cc')^2 m \log^2 n)$.

Assume that the output of the algorithm is YES. Then it follows from Lemma 11 that there exist two points x and y on P such that $x \leq_P y$ and $\delta_F(Q, P[x, y]) \leq 3\Delta$.

On the other hand, assume there exist two points x and y on P such that $x \leq_P y$ and $\delta_F(Q, P[x, y]) < \Delta$. Consider the points $x'_1, y'_1, \ldots, x'_{m-1}, y'_{m-1}$ that satisfy the five properties of Lemma 10. We observe that, for each i with $1 \leq i < m$, the pair (x'_i, y'_i) is contained in the set K_i. Using the third property of Lemma 10 and Lemma 12, it then follows that for each i with $1 \leq i < m$, the pair (x'_1, y'_i) is in the set K'_i. In particular, at the end of the query algorithm, the pair (x'_1, y'_{m-1}) is in the set K'_{m-1}. Thus, since this set is non-empty, the query algorithm returns YES.

For this result, we can take ε to be a sufficiently small constant. Thus, assuming that P is c-packed and each of its edges has lenght at least Δ/c', we obtain a data structure with $O(n \log^2 n + cn \log n)$ preprocessing time, $O(n \log n + cn)$ size, and $O((cc')^2 m \log^2 n)$ query time.

To remove the assumption that each edge of P has length at least Δ/c', we first compute a μ-simplification P' of P, with $\mu = \Delta/c'$, as we did in Section 4.3. Then we build the above data structure for the simplified path P'. For any given real number $\varepsilon' > 0$, if we take $c' = 1/\varepsilon'$, we obtain the following result[3].

Theorem 3 *Let P be a polygonal path in the plane with n vertices, let c and Δ be positive real numbers, and assume that P is c-packed. For any $\varepsilon > 0$, we can construct a data structure of size $O(n \log n + cn)$ in $O(n \log^2 n + cn \log n)$ time. Given any polygonal path Q with m vertices, all of whose edges have length more than 4Δ, the query algorithm corresponding to this data structure takes $O((c/\varepsilon)^2 m \log^2 n)$ time and outputs either YES or NO.*

[3]In the theorem, we rename ε' as ε.

1. *If the output is YES, then there exist two points x and y on P with $x \leq_P y$ such that $\delta_F(Q, P[x,y]) \leq 3(1+\varepsilon)\Delta$.*

2. *If the output is NO, then $\delta_F(Q, P[x,y]) \geq \Delta$ for any two points x and y on P with $x \leq_P y$.*

6.2 Querying a Tree with a Polygonal Path

We finally consider a tree T with n vertices. Let Δ, c, and c' be positive real numbers, and assume that T is c-packed and each of its edges has length at least Δ/c'.

We choose a constant $\varepsilon > 0$ and preprocess T as in Section 5.2. Additionally, for each path P in the path decomposition $PD(T)$, we construct the data structure $SFP(P)$ of Section 6.1. For any disk D in the plane (given by its center and radius) and any two points x and y on T, we can use $PD(T)$ and the structures $SFP(P)$ to decide, in $O(\log^3 n)$ time, if the path $T[x,y]$ is completely contained in D.

The entire preprocessing algorithm takes $O(n \log^2 n + cn \log n)$ time and the resulting data structure has size $O(n \log n + cn)$.

Let $Q = (q_1, q_2, \ldots, q_m)$ be a query path all of whose edges have length more than 4Δ. For each integer i with $1 \leq i < m$, we run Steps 1, 2, and 3 of the query algorithm of Section 5.3 with the segment $[q_i, q_{i+1}]$. This gives the set J_i of all pairs (x, y) of points such that

- x is an intersection point between T and the half-circle $C_{q_i q_{i+1}}$,
- y is an intersection point between T and the half-circle $C_{q_{i+1} q_i}$,
- the path $T[x,y]$ is completely contained in the region $R(q_i, q_{i+1}) \setminus (D_{q_i} \cup D_{q_{i+1}})$.

We change Step 4 of the algorithm in Section 5.3, again with the segment $[q_i, q_{i+1}]$, as follows. We consider all pairs (x, y) in J_i. For each such pair, if the test in Step 4 is positive, then we insert this pair into an initially empty set K_i.

At the end of Step 4, K_i is the set of all pairs (x, y) satisfying the three properties in Lemma 3, with respect to the line segment $[q_i, q_{i+1}]$.

After initializing $K'_1 = K_1$, we do the following for $i = 1, 2, \ldots, m-2$: Set $K'_{i+1} = \emptyset$. Then consider each pair in K'_i. For each such pair (x, y), find all pairs (x', y') in K_{i+1} such that (i) y is on the path $T[x, x']$, (ii) x' is on the path $T[y, y']$, and (iii) the path $T[y, x']$ is completely contained in the disk with center q_{i+1} and radius 3Δ. For each such pair (x', y') found, insert the pair (x, y') into the set K'_{i+1}. (It follows from Lemma 12 that (x, y') was not in K'_{i+1} yet.) At the end of this algorithm, we have obtained a set K'_{m-1}. If this set is non-empty, then we return YES; otherwise, we return NO.

Before we analyze the query time, observe that each of the sets K_i has size $O((cc')^2)$. The set K'_i contains pairs (x, y) of points on T with x on $C_{q_1 q_2}$ and y on $C_{q_{i+1} q_i}$. Therefore, the size of K'_i is $O((cc')^2)$ as well.

The total time for Steps 1, 2, and 3 is $O((cc')^2 m \log^3 n + (cc')^3 m \log n)$. The modified Step 4 takes $O((cc')^2 \log^2 n)$ time. The time to compute the set K'_{i+1} is $O(|K'_i| \cdot |K_{i+1}| \log^3 n)$, which is $O((cc')^4 \log^3 n)$. It follows that the total query time is $O((cc')^4 m \log^3 n)$.

If the output of the query algorithm is YES, then it follows from Lemma 11 that there exist two points x and y on T such that $\delta_F(Q, T[x, y]) \leq 3\Delta$.

Assume there exist two points x and y on T such that $\delta_F(Q, T[x, y]) < \Delta$. Consider the points $x'_1, y'_1, \ldots, x'_{m-1}, y'_{m-1}$ that satisfy the five properties of Lemma 10. For each integer i with $1 \leq i < m$, the pair (x'_i, y'_i) is contained in the set K_i. Using the third property of Lemma 10 and Lemma 12, it follows that for each i with $1 \leq i < m$, the pair (x'_1, y'_i) is in the set K'_i. Thus, at the end of the query algorithm, the pair (x'_1, y'_{m-1}) is in the set K'_{m-1}. As a result, the query algorithm returns YES.

We have proved our final result:

Theorem 4 *Let T be a tree with n vertices that is embedded in the plane, and let c, c', and Δ be positive real numbers. Assume that T is c-packed and each of its edges has length at least Δ/c'. We can construct a data structure of size $O(n \log n + cn)$ in $O(n \log^2 n + cn \log n)$ time. Given any polygonal path Q with m vertices, all of whose edges have length more than 4Δ, the query algorithm corresponding to this data structure takes $O((cc')^4 m \log^3 n)$ time and outputs either YES or NO.*

1. *If the output is YES, then there exist two points x and y on T such that $\delta_F(Q, T[x, y]) \leq 3\Delta$.*

2. *If the output is NO, then $\delta_F(Q, T[x, y]) \geq \Delta$ for any two points x and y on T.*

7 Concluding Remarks

We have presented data structures that, for a fixed real number $\Delta > 0$, store a geometric tree T such that, for any given polygonal query path Q, we can approximately decide if the tree contains a path whose Fréchet distance to Q is less than Δ. We obtained good bounds on the preprocessing time, size, and query time for trees that are c-packed, for some small value of c, and for queries Q in which each edge has length $\Omega(\Delta)$. For general trees, we also need that each of its edges has length $\Omega(\Delta)$; for the case when T is a polygonal path, the latter requirement is not needed. We leave as an open problem to remove this requirement for general trees.

Our approximation factors are $\sqrt{2}(1+\varepsilon)$ if Q is a query segment, and 3 if Q is a polygonal path. We leave as an open problem to improve these approximation factors.

All our results assume that the value Δ is fixed. It would be interesting to obtain an efficient data structure for which Δ is part of the query.

References

[1] P. K. Agarwal, R. B. Avraham, H. Kaplan, and M. Sharir. Computing the discrete Fréchet distance in subquadratic time. In *Proceedings of the 24th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 156–167. SIAM, 2013.

[2] H. Alt. The computational geometry of comparing shapes. In *Efficient Algorithms, Essays Dedicated to Kurt Mehlhorn on the Occasion of His 60th Birthday*, pages 235–248. Springer, 2009.

[3] H. Alt, A. Efrat, G. Rote, and C. Wenk. Matching planar maps. *Journal of Algorithms*, 49(2):262–283, 2003.

[4] H. Alt and M. Godau. Computing the Fréchet distance between two polygonal curves. *International Journal of Computational Geometry & Applications,*, 5:75–91, 1995.

[5] H. Alt, C. Knauer, and C. Wenk. Comparison of distance measures for planar curves. *Algorithmica*, 38(1):45–58, 2003.

[6] S. Brakatsoulas, D. Pfoser, R. Salas, and C. Wenk. On map-matching vehicle tracking data. In *31st Conference on Very Large Data Bases (VLDB)*, pages 853–864. VLDB Endowment, 2005.

[7] K. Buchin, M. Buchin, and J. Gudmundsson. Constrained free space diagrams: a tool for trajectory analysis. *Int. Journal of GIS*, 24(7):1101–1125, 2010.

[8] K. Buchin, M. Buchin, C. Knauer, G. Rote, and C. Wenk. How difficult is it to walk the dog? *23rd European Workshop on Computational Geometry (EuroCG)*, pages 170–173, 2007. Graz, Austria.

[9] K. Buchin, M. Buchin, W. Meulemans, and W. Mulzer. Four soviets walk the dog - with an application to Alt's conjecture. *CoRR*, abs/1209.4403, 2012.

[10] D. Chen, A. Driemel, L. J. Guibas, A. Nguyen, and C. Wenk. Approximate map matching with respect to the Fréchet distance. In *Proceedings of the Workshop on Algorithm Engineering and Experiments (ALENEX)*, pages 75–83. SIAM, 2011.

[11] S.-W. Cheng, O. Cheong, H. Everett, and R. van Oostrum. Hierarchical decompositions and circular ray shooting in simple polygons. *Discrete & Computational Geometry*, 32:401–415, 2004.

[12] R. Cole and U. Vishkin. The accelerated centroid decomposition technique for optimal parallel tree evaluation in logarithmic time. *Algorithmica*, 3:329–346, 1988.

[13] M. de Berg, A. F. Cook IV, and J. Gudmundsson. Fast Fréchet queries. *Computational Geometry – Theory and Applications*, 46(6):747–755, 2013.

[14] A. Driemel, S. Har-Peled, and C. Wenk. Approximating the Fréchet distance for realistic curves in near linear time. *Discrete & Computational Geometry*, 48:94–127, 2012.

[15] M. Fréchet. Sur quelques points du calcul fonctionnel. *Rendiconti del Circolo Matematico di Palermo*, 22:1–74, 1906.

[16] E. J. Keogh and M. J. Pazzani. Scaling up dynamic time warping to massive datasets. In *Proceedings of the 3rd European Conference on Principles and Practice of Knowledge Discovery in Databases*, pages 1–11. Springer, 1999.

[17] E. M. McCreight. Priority search trees. *SIAM Journal on Computing*, 14:257–276, 1985.

[18] C. A. Ratanamahatana and E. J. Keogh. Three myths about dynamic time warping data mining. In *Proceedings of the 5th SIAM International Data Mining Conference*. SIAM, 2005.

[19] E. Sriraghavendra, K. Karthik, and C. Bhattacharyya. Fréchet distance based approach for searching online handwritten documents. In *Proceedings of the 9th International Conference on Document Analysis and Recognition*, pages 461–465, 2007.

[20] C. Wenk. Shape matching in higher dimensions; chapter 5: Matching curves with respect to the Fréchet distance. *Dissertation, Freie Universität Berlin, Germany*, 2003.

[21] C. Wenk, R. Salas, and D. Pfoser. Addressing the need for map-matching speed: Localizing global curve-matching algorithms. In *Proceedings of the 18th Conference on Scientific and Statistical Database Management (SSDBM)*, pages 379–388, 2006.