Farthest-Point Queries with Geometric and Combinatorial Constraints

Ovidiu Daescu* Ningfang Mi† Chan-Su Shin‡ Alexander Wolff§

Proximity problems are fundamental in computational geometry and have been studied intensively since Knuth [3] has posed the post-office problem about three decades ago. In this paper we discuss farthest-point problems in which a sequence S of n points in the plane is given in advance and can be preprocessed to answer various queries efficiently. Our main results are the following.

First, we present a data structure that can be used to compute the point farthest from a query line segment in $O(\log^2 n)$ time. Our data structure needs $O(n \log n)$ space and preprocessing time. To the best of our knowledge no solution to this problem has been suggested yet.

Second, we design a data structure that can be used to simplify polygonal paths in the following sense: given a path $P = (p_1, \ldots, p_n)$ and a real $\Delta > 0$ we want to find a subpath P' of P that consists exclusively of Δ-*approximating segments* according to the *tolerance-zone criterion*, i.e. line segments $\overline{p_i p_k}$ with the property that each p_j with $i < j < k$ is at most Δ away from $\overline{p_i p_k}$. We are interested in a min-# subpath, i.e. a subpath with the minimum number of vertices. This is motivated by data reduction and considered an important problem—finding a near-linear solution is listed as problem 24 in the *Open Problems Project* [4]. Our query-based algorithm finds a min-# subpath in $O(n^2 \log^3 n)$ worst-case running time. This is slightly worse than the quadratic running time of the best incremental algorithm [1], but much better in practice since, as we will see later (Theorem 2), the running time of our algorithm is output sensitive. Our algorithm has the same structure as a query-based algorithm [2] for the weaker *infinite-beam criterion* which requires that a vertex p_j of P that is shortcut by an edge $\overline{p_i p_k}$ of P' must be within Δ from the *line* $p_i p_k$ through p_i and p_k. The algorithm [2] outperformed an incremental algorithm similar to [1] in an experimental evaluation.

Both main results are based on our solution of the problem *FarthestVertexInHalfplane* (FV-halfplane): Preprocess a convex n-gon C for queries of the following type. Given (q, l_q), where q is a point and l_q is a directed line through q, decide whether there is a vertex of C to the left of l_q. If yes report the one farthest from q. (See Figure 1.)

Other than one might think at first glance, this problem cannot be solved simply by binary search on the vertices of C since the distance from the query point q is not unimodular on the boundary of C. Binary search does help us, however, to determine which part of C lies to the left of l_q. We construct a binary tree whose leaves are the vertices of C. Each interior node v is labeled by the set C_v of vertices of C in the leaves below v. Node v stores the farthest-point Voronoi diagram of C_v. The part of C to the left of l_q is covered by the labels of $O(\log n)$ interior nodes. We locate the query point q in the corresponding diagrams. Among the points in whose cells q lies, we return the one farthest from q. This yields:

Lemma 1 *There is a data structure for FV-halfplane that takes $O(n \log n)$ space, $O(n \log n)$ preprocessing time and $O(\log^2 n)$ query time.*

Next we address a problem whose solution yields our first main result, a structure for finding points farthest from query line segments.

FarthestPointInHalfstrip (FP-halfstrip): Preprocess a set S of n points for queries of the following type. Given a triplet (q, l_q, Δ), where q is a point and l_q is a directed line through q such that all points in S are within distance Δ

*Department of Computer Science, University of Texas at Dallas, Richardson, TX 75083, USA. Email: daescu@utdallas.edu

†Department of Computer Science, College of William and Mary, P.O. Box 8795, Williamsburg, VA 23187-8795, USA. Email: ningfangmi@hotmail.com

‡School of Electronics and Information Engineering, Hankuk University of Foreign Studies, Korea. Email: cssin@hufs.ac.kr

§Faculty of Computer Science, Karlsruhe University, P.O. Box 6980, D-76128 Karlsruhe, Germany. WWW: i11www.ilkd.uka.de/people/awolff

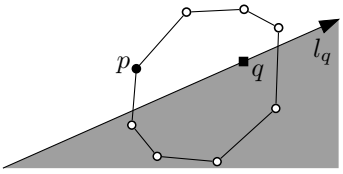

Figure 1: FV-halfplane.

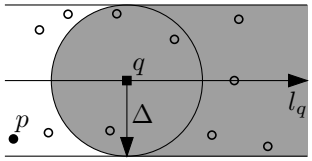

Figure 2: FP-halfstrip.

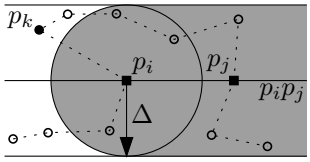

Figure 3: FIP-halfstrip.

from l_q, decide whether there is a point $p \in S$ such that (i) $|qp| > \Delta$, and (ii) the projection of p on l_q lies before q. If yes, report the point farthest from q that fulfills (i) and (ii). (See Figure 2.)

We prove that if there are points fulfilling (i) and (ii), then the one farthest from q among them lies on the convex hull of S. (Note that this is not true if we drop (i).) Thus our data structure for FV-halfplane in fact solves FP-halfstrip within the same asymptotic bounds. Now we can state our first main result:

Theorem 1 *We can preprocess a set S of n points in $O(n \log n)$ time and space such that the point in S farthest from a query line segment s can be reported in $O(\log^2 n)$ time.*

Proof. Let l_s be the line supporting s. The point farthest from s is either (i) the point farthest from l_s or (ii) the point p farthest from one of the endpoints of s with the property that the line orthogonal to s at that endpoint has s and p on different sides. The point for (i) can be obtained in $O(\log n)$ time by binary search on the convex hull of S, since the distance from l_s is unimodular on each of the at most two curves into which l_s splits the hull. This gives us a value Δ such that all points in S are within distance Δ from l_s. The point for (ii) can then be found by performing two queries, one for each endpoint of s, in the data structure for FP-halfstrip. □

In polygonal paths the point order is important. Therefore we now consider an indexed version of FP-halfstrip, namely *FarthestIndexedPointInHalfstrip* (FIP-halfstrip): Preprocess a sequence $S = (p_1, \ldots, p_n)$ of points for queries of the following type. Given a triplet (i, j, Δ) such that all points p_k with $i < k < j$ are within distance Δ from the line p_ip_j, decide whether there is a point p_k with $i < k < j$ such that (i) $|p_ip_k| > \Delta$, and (ii) p_j and the projection of p_k on p_ip_j lie on different sides of p_i. If yes, report the point p_k farthest from p_i that fulfills (i) and (ii). (See Figure 3.)

We solve FIP-halfstrip using a binary tree of the same structure as that for FV-halfplane, except that in each interior node we do not store a farthest-point Voronoi diagram but the data structure for FV-halfplane for the corresponding subset of S. Our time and space bounds for FIP-halfstrip are by a log-factor above those for FV-halfplane. The data structure for FIP-halfstrip yields an output-sensitive query-based algorithm for polygonal path simplification.

Theorem 2 *Given a polygonal path $P = (p_1, \ldots, p_n)$ in $\mathbb{R}^2$ and a real $\Delta > 0$, a subpath P' of P with the minimum number m of vertices among all subpaths satisfying the tolerance-zone criterion can be computed in $O(F(m)\, n \log^3 n)$ time using $O(n \log^2 n)$ space, where $F(m) \in O(n)$ is the number of vertices that can be reached from p_1 with at most $(m-2)$ Δ-approximating segments.*

The algorithm is similar to [2], except that each query takes $O(\log^3 n)$ time instead of $O(\log n)$ time. We spend $O(\log n)$ time to decide whether some $\overline{p_ip_j}$ is a Δ-approximating segment according to the infinite-beam criterion. If the answer is yes, we query the FIP-halfstrip data structure in $O(\log^3 n)$ time to decide whether $\overline{p_ip_j}$ is also a Δ-approximating segment according to the tolerance-zone criterion.

References

[1] D. Chen and O. Daescu. Space-efficient algorithms for approximating polygonal curves in two dimensional space. *Int. Jour. Comp. Geom. Appl.*, 13(2):95–112, 2003.

[2] O. Daescu and N. Mi. Polygonal path approximation: A query based approach. In *Proc. ISAAC'03*, volume 2906 of *LNCS*, pages 36–46. Springer-Verlag, Dec. 2003.

[3] D. E. Knuth. *The Art of Computer Programming*, volume 3, chapter Sorting and Searching. Addison-Wesley, Reading, MA, 1973.

[4] J. S. B. Mitchell and J. O'Rourke. Computational geometry column 42. *SIGACT News*, 32(3):63–72, 2001.