Data Structures for Halfplane Proximity Queries and Incremental Voronoi Diagrams

Boris Aronov* Prosenjit Bose† Erik D. Demaine‡ Joachim Gudmundsson§
John Iacono¶ Stefan Langerman‖ Michiel Smid†

November 22, 2005

Abstract

We consider preprocessing a set S of n points in the plane that are in convex position into a data structure supporting queries of the following form: given a point q and a directed line ℓ in the plane, report the point of S that is farthest from (or, alternatively, nearest to) the point q subject to being to the left of line ℓ. We present two data structures for this problem. The first data structure uses $O(n^{1+\varepsilon})$ space and preprocessing time, and answers queries in $O(2^{1/\varepsilon} \log n)$ time. The second data structure uses $O(n \log^3 n)$ space and polynomial preprocessing time, and answers queries in $O(\log n)$ time. These are the first solutions to the problem with $O(\log n)$ query time and $o(n^2)$ space.

In the process of developing the second data structure, we develop a new representation of nearest-point and farthest-point Voronoi diagrams of points in convex position. This representation supports insertion of new points in counterclockwise order using only $O(\log n)$ amortized pointer changes, subject to supporting $O(\log n)$-time point-location queries, even though every such update may make $\Theta(n)$ combinatorial changes to the Voronoi diagram. This data structure is the first demonstration that deterministically and incrementally constructed Voronoi diagrams can be maintained in $o(n)$ pointer changes per operation.

1 Introduction

Line simplification is an important problem in the area of digital cartography [Cro91, Den98, MS92]. Given a polygonal chain P, the goal is to compute a simpler polygonal chain Q that provides a good approximation to P. Many variants of this problem arise depending on how one defines *simpler* and how one defines *good approximation*. Almost all of the known methods of approximation compute distances between P and Q. Therefore, preprocessing P in order to quickly answer distance queries is a common subproblem to most line simplification algorithms.

Of particular interest to our work is a line simplification algorithm proposed by Daescu et al. [DMSW05]. Given a polygonal chain $P = (p_1, p_2, \ldots, p_n)$, they show how to compute a subchain $P' = (p_{i_1}, p_{i_2}, \ldots, p_{i_m})$, with $i_1 = 1$ and $i_m = n$, such that each segment $[p_{i_j} p_{i_{j+1}}]$ of P' is a good approximation of the subchain of P from p_{i_j} to $p_{i_{j+1}}$. The amount of error is determined by the point of the subchain that is farthest from the line segment $[p_{i_j} p_{i_{j+1}}]$. To compute this approximation efficiently, the key subproblem they solve is the following:

*Department of Computer and Information Science, Polytechnic University, Brooklyn, NY, USA. Research supported in part by NSF grant ITR-0081964 and by a grant from US-Israel Binational Science Foundation.

†School of Computer Science, Carleton University, Ottawa, ON, Canada. Research supported in part by NSERC.

‡Computer Science and Artificial Intelligence Laboratory, MIT, Cambridge, MA, USA. Research supported in part by NSF grants CCF-0430849 and OISE-0334653.

§National ICT Australia, Sydney, Australia.

¶Department of Computer and Information Science, Polytechnic University, Brooklyn, NY, USA. Research supported in part by NSF grants CCF-0430849 and OISE-0334653.

‖Chercheur qualifié du FNRS, Départment d' Informatique, Université Libre de Bruxelles, Brussels, Belgium.

Problem 1 (HFPQ: Halfplane Farthest-Point Queries). *Preprocess a set $S = \{p_1, p_2, \ldots, p_n\}$ of n points in the plane that are in convex position into a data structure supporting the following query: given a point q and a directed line ℓ in the plane, report the point of S that is farthest from q subject to being to the left line ℓ.*

Daescu et al. [DMSW05] show that, with $O(n \log n)$ preprocessing time and space, these queries can be answered in $O(\log^2 n)$ time. On the other hand, a naïve approach achieves $O(\log n)$ query time by using $O(n^3)$ preprocessing time and $O(n^3)$ space. The open question they posed is whether $O(\log n)$ query time can be obtained with a data structure using subcubic and preferably subquadratic space.

In this paper, we solve this problem with two data structures. The first, relatively simple data structure uses $O(n^{1+\varepsilon})$ preprocessing time and space, and answers queries in $O(2^{1/\varepsilon} \log n)$ time. The second, more sophisticated data structure uses $O(n \log^3 n)$ space and polynomial preprocessing time, and answers queries in $O(\log n)$ time. Both of our data structures apply equally well to halfplane farthest-point queries, described above, as well as the opposite problem of halfplane nearest-point queries. Together we refer to these queries as *halfplane proximity queries*.

Dynamic Voronoi diagrams. An independent contribution of the second data structure is that it provides a new efficient representation for maintaining the nearest-point for farthest-point Voronoi diagram of a dynamic set of points. So far, point location in dynamic planar Voronoi diagrams has proved difficult because the complexity of the changes to the Voronoi diagram or Delaunay triangulation for an insertion can be linear at any one step. The randomized incremental construction avoids this worst-case behavior through randomization. However, for the deterministic insertion of points, the linear worst-case behavior cannot be avoided, even if the points being incrementally added are in convex position, and are added in order (say, counterclockwise). For this specific case, we give a representation of a (nearest-point or farthest-point) Voronoi diagram that supports $O(\log n)$-time point location in the diagram while requiring only $O(\log n)$ pointer changes in the structure for each update. So as not to oversell this result, we note that we do not have an efficient method of determining which pointers to change (it takes $\Theta(n)$ time per change), so the significance of this representation is that it serves as a proof of the existence of an encoding of Voronoi diagrams that can be modified with few changes to the encodings while still supporting point location queries. However, we believe that the combinatorial observations on Voronoi diagrams contained herein will help lead to efficient dynamic Voronoi diagrams.

Acknowledgments

This work was initiated at the Schloß Dagstuhl Seminar 04091 on Data Structures, organized by Susanne Albers, Robert Sedgewick, and Dorothea Wagner, and held February 22–27, 2004 in Germany. This work continued at the Korean Workshop on Computational Geometry and Geometric Networks, organized by Hee-Kap Ahn, Christian Knauer, Chan-Su Shin, Alexander Wolff, and René van Oostrum, and held July 25–30, 2004 at Schloß Dagstuhl in Germany; and at the 2nd Bertinoro Workshop on Algorithms and Data Structures, organized by Andrew Goldberg and Giuseppe Italiano, and held May 29–June 4, 2005 in Italy. We thank the organizers and institutions hosting both workshops for providing a productive research atmosphere. We also thank Alexander Wolff for introducing the problem to us.

References

[Cro91] Robert G. Cromley. *Digital Cartography.* Prentice Hall, August 1991.

[Den98] Borden D. Dent. *Cartography: Thematic Map Design.* William C Brown Pub, fifth edition, July 1998.

[DMSW05] Ovidiu Daescu, Ningfang Mi, Chan-Su Shin, and Alexander Wolff. Farthest-point queries with geometric and combinatorial constraints. *Computational Geometry: Theory and Applications*, 2005. To appear.

[MS92] Robert B. McMaster and K. Stuart Shea. *Generalization in Digital Cartography.* Association of American Cartographers, Washington D.C., 1992.