Information Processing Letters 42 (1992) 55–60
North-Holland 27 April 1992

Optimal time bounds for some proximity problems in the plane

Alok Aggarwal
IBM Research Division, Thomas J. Watson Research Center, Yorktown Heights, NY 10598, USA

Herbert Edelsbrunner *
Department of Computer Science, University of Illinois at Urbana-Champaign, Urbana, IL 61801, USA

Prahakar Raghavan and Prasoon Tiwari
IBM Research Division, Thomas J. Watson Research Center, Yorktown Heights, NY 10598, USA

Communicated by D.A. Plaisted
Received 8 November 1989
Revised 9 August 1990

Abstract

Aggarwal, A., H. Edelsbrunner, P. Raghavan and P. Tiwari, Optimal time bounds for some proximity problems in the plane, Information Processing Letters 42 (1992) 55–60.

Given a sequence of n points that form the vertices of a simple polygon, we show that determining a closest pair requires $\Omega(n \log n)$ time in the algebraic decision tree model. Together with the well-known $O(n \log n)$ upper bound for finding a closest pair, this settles an open problem of Lee and Preparata. We also extend this $O(n \log n)$ upper bound to the following problem: Given a collection of sets with a total of n points in the plane, find for each point a closest neighbor that does not belong to the same set.

Keywords: Computational geometry, upper and lower bounds, point sets, simple polygons, Euclidean distance, algebraic decision tree model, integer element uniqueness problem

1. Introduction

The *closest-pair problem* for a set S of n points in the plane is to find two points $p,q \in S$ so that

$$d(p, q) = \min_{r,s \in S} \{d(r, s)\},$$

where $d(p, q)$ denotes the Euclidean distance between p and q. It is well known that this problem can be solved in $O(n \log n)$ time, see e.g. [5]. Shamos [15] has also shown that this problem requires $\Omega(n \log n)$ time in the linear decision tree model, and a general technique due to Ben-Or [4] can be used to extend the $\Omega(n \log n)$ lower bound to the algebraic decision tree model. The lower bound is obtained by reducing the closest-pair problem to the element uniqueness problem for a set of $\Theta(n)$ real numbers, and it does not apply when the points are given as a sequence

Correspondence to: Dr. A. Aggarwal, IBM Research Division, Thomas J. Watson Research Center, P.O. Box 218, Yorktown Heights, NY 10598, USA. Email: aggarwa@watson.ibm.com.

* Research supported by the National Science Foundation under grant CCR-8714565.

that defines a simply polygon [1]. If the sequence forms a convex polygon then $O(n)$ times suffices to find the closest pair, see [10]. Indeed, they showed that $O(n)$ time suffices to compute a closest neighbor [2] for each point.

The discrepancy between the $\Omega(n \log n)$ lower bound and the $O(n)$ upper bound motivated Lee and Preparata [11] to ask whether $\Omega(n \log n)$ is a lower bound for the closest-pair problem even if the sequence of points defines a simply polygon. In this paper, we settle this question with the affirmative and show the somewhat stronger result that $\Omega(n \log n)$ is required even when the points define a monotone or a star-shaped polygon [3]. We complement this lower bound with an $O(n \log n)$ upper bound for the following more general proximity problem.

> Given a collection of point sets, $S_1, S_2, \ldots, S_k$, with a total of n points, for each point find a point that is closest but not in the same set.

The algorithm that solves the problem in time $O(n \log n)$ uses Voronoi diagrams of various subsets of $S = \bigcup_{1 \leqslant i \leqslant k} S_i$ (see [7] or [13] for extensive treatments of Voronoi diagrams). It extends the $O(n \log n)$ upper bounds for the closest-pair problem and the all closest neighbors problem [4] for a point set.

The organization of this paper is as follows. Section 2 demonstrates the $\Omega(n \log n)$ lower bound. This is done by a reduction to the so-called integer element uniqueness problem for which Yao [16] has recently shown an $\Omega(n \log n)$ lower bound in the algebraic decision tree model. Section 3 gives the upper bound for the closest neighbors problem of a collection of point sets. In Section 4 we briefly address an implication of Yao's lower bound on the difference between the RAM model and the algebraic decision tree model of computation. Finally, Section 5 concludes with some remarks and open problems.

2. The closest-pair problem – a lower bound

This section demonstrates that the closest-pair problem for a sequence of n points in the plane takes $\Omega(n \log n)$ time even if the sequence forms a simple polygon. The proof uses a recent $\Omega(n \cdot \log n)$ lower bound of Yao [16] for the *integer element uniqueness problem*. This problem asks whether or not any two of a given sequence of n integers are equal. The lower bound holds in the algebraic decision tree model.

Theorem 2.1. *Given the sequence of n vertices of a monotone or star-shaped polygon, to determine the closest pair of vertices requires $\Omega(n \log n)$ time in the algebraic decision tree model.*

Proof. We show that if the closest-pair problem for the vertices of a monotone or star-shaped polygon with $n+2$ vertices can be solved in time $T(n)$, then the integer element uniqueness problem for n integers can be solved in time $O(T(n)+n)$. This coupled with the $\Omega(n \log n)$ lower bound given by Yao [16] yields the desired result.

Given a sequence $Y = (y_1, y_2, \ldots, y_n)$ of integers, we construct a polygon P with $n+2$ vertices $p_0, p_1, \ldots, p_{n+1}$. For $1 \leqslant i \leqslant n$ define $p_i = (x_i, y_i)$ with $x_i = i/(n+1)$. It is enough to choose $x_0 = 0$, $x_{n+1} = 1$ and the y-coordinates of p_0 and p_{n+1} smaller than any of the y_i to complete P so that it is monotone. However, to guarantee that it is also star-shaped we have to be slightly more sophisticated in the choice of y_0 and y_{n+1}. Define y' as the minimum y-coordinate of any intersection point of the vertical lines $x = 0$ and

[1] We obtain the *polygon* formed by the sequence $(p_1, p_2, \ldots, p_n)$ when we connect p_i with p_{i+1} by a straight line edge, for $1 \leqslant i \leqslant n$ and $p_{n+1} = p_1$. The polygon is *simple* if no two edges intersect, except for adjacent edges which meet in a common endpoint.

[2] A *closest neighbor* of a point $p \in S$ is a point $q \in S - \{p\}$ so that $d(p, q) = \min_{r \in S}\{d(p, r)\}$.

[3] A simple polygon is *monotone* if there is a direction so that any line parallel to this direction meets the polygon in at most two points. It is *star-shaped* if there is a point inside the polygon so that every line through this point meets the polygon in exactly two points. The set of such points is called the *kernel* of the polygon.

[4] The *all closest neighbors problem* for a finite set of points in the plane asks for a closest neighbor for each point in the set.

$x=1$ with the lines defined by any two consecutive points of $p_1, p_2, \ldots, p_n$. That is,

$$y' = \min_{1 \leqslant i \leqslant n} \{ y_i - i(y_{i+1} - y_i), \\ y_{i+1} - (n-i)(y_i - y_{i+1}) \}.$$

For technical reasons that will become apparent soon we choose $p_0 = (0, y_0)$ and $p_{n+1} = (1, y_{n+1})$ with y_0 and y_{n+1} even smaller than y'. Define $\delta = \max_{1 \leqslant i,j \leqslant n} \{ y_i - y_j \}$ and set

$$y_0 = y' - (\delta + 1) \quad \text{and} \quad y_{n+1} = y' - 2(\delta + 1).$$

At this point it can be readily seen that $p_0, p_1, \ldots, p_{n+1}$ form a simple polygon that can be constructed in linear time from Y; P is monotone (any vertical line meets it in at most two points) and star-shaped (any point slightly above the edge $p_0 p_{n+1}$ lies in its kernel). Below we show that if the closest-pair problem for P can be solved in $T(n)$ time steps then the integer element uniqueness problem for Y can be solved in time $O(T(n) + n)$.

Suppose an algorithm $\mathscr{A}$ determines a closest pair, $\{p_k, p_\ell\}$ with $k < \ell$, in $T(n)$ time steps. By the choice of p_0 and p_{n+1} we have $k \neq 0$ and $\ell \neq n+1$. Set

$$d_{k,\ell} = \sqrt{(x_k - x_\ell)^2 + (y_k - y_\ell)^2},$$

the distance between p_k and p_ℓ. We claim that there are two integers in Y that are equal if and only if $d_{k,\ell} \leqslant 1$. To prove the claim, note that if y_i and y_j denote any two integers in Y and if $y_i = y_j$, then the corresponding distance between points p_i and p_j is $|i-j|/(n+1) < 1$. On the other hand, if all integers in Y are distinct, then the minimum distance between any two vertices of P is strictly greater than 1. Thus, given the closest pair of P we can decide the element uniqueness problem for Y in constant time. □

Remarks. (1) Notice that the vertices of the monotone polygon in the above proof occur in sorted order along the x-direction. Hence, Theorem 2.1 implies an $\Omega(n \log n)$ lower bound for the closest-pair problem even when the points are given in sorted order from left to right. In comparison, the furthest pair of a sequence of n points that form a simple polygon can be found in time $O(n)$. This is because the convex hull of the polygon can be computed in time $O(n)$ (see e.g. [8,12]) and the diameter of the resulting convex polygon, determined by the furthest pair, can be found in time $O(n)$ (see e.g. [13]).

(2) A result similar to Theorem 2.1 is the $\Omega(n \log n)$ lower bound for constructing the Delaunay triangulation of n points sorted along the x-direction due to Seidel [14]. Since every closest pair in the point set forms an edge of the Delaunay triangulation, Theorem 2.1 can be used to obtain the same result. However, it is unclear whether the techniques in [14] can be modified to establish the result of Theorem 2.1.

3. The closest neighbors problem – an upper bound

In this section we extend the well-known $O(n \cdot \log n)$ upper bound for the all closest neighbors problem for n points in the plane to the more general setting where the n points come in $k \leqslant n$ sets, $S_1, S_2, \ldots, S_k$. For each point p we need to find a point in $S = \bigcup_{1 \leqslant i \leqslant k} S_i$ that is closest to p but not in the same set as p.

We use Voronoi diagrams and point location algorithms to solve the problem. We review the basic properties of both and refer to [13] and [7] for further details.

The Voronoi region of a point q in some point set P is the set of all points x in the plane that are closer to q than to any other point in P. The *Voronoi diagram* of P, $\mathscr{V}(P)$, is the subdivision whose regions are the Voronoi regions of the points in P. Two points in P are said to be *Voronoi neighbors* if their regions have a common edge. Using Euler's relation for planar graphs it is straightforward to prove that the number of Voronoi neighbor pairs is at most $3|P| - 6$ if $|P| \geqslant 3$. To *locate* a point x in $\mathscr{V}(P)$ means to determine a region of $\mathscr{V}(P)$ whose closure contains x. The point of P that generates this region is a closest neighbor of x. There are data structures that allow a point to be located in time $O(\log n)$, after $O(n \log n)$ time for preprocessing (see e.g. [7,13]).

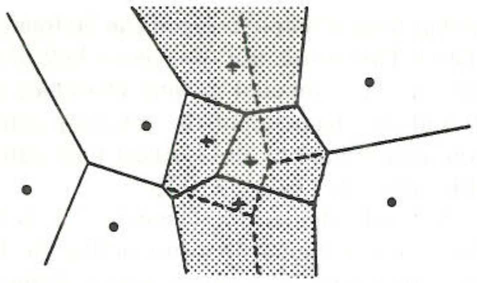

Fig. 1. Voronoi diagram of points in the plane with the domain of one set (marked with "+") further decomposed by the Voronoi diagram of the other points. Note that a "+" point can have a closest neighbor which is not a Voronoi neighbor.

Notice that it is enough to locate every point $p \in S_i$ in the Voronoi diagram of $S - S_i$, but constructing all k diagrams $\mathscr{V}(S - S_i)$ takes much more than $O(n \log n)$ time unless k is very small. We thus come up with a refinement of this approach (see Fig. 1):

```
Step 1. Construct $\mathscr{V}(S)$, the Voronoi diagram of
        $S$.
Step 2. for i := 1 to $k$ do
            compute $T_i = \{p \in S - S_i \mid p$ has a
            Voronoi neighbor in $S_i\}$;
            construct $\mathscr{V}(T_i)$;
            locate the points of $S_i$ in $\mathscr{V}(T_i)$;
        endfor.
```

To prove the correctness of this algorithm we just need to show that if $q \notin S_i$ is a closest neighbor of a point $p \in S_i$, then q is Voronoi neighbor of some point in S_i. Notice however that q is not necessarily a Voronoi neighbor of p itself. To see that q is a Voronoi neighbor of some point in S_i compare $\mathscr{V}(S - S_i)$ and $\mathscr{V}(S)$. The two Voronoi diagrams agree outside $\mathscr{W}_i$ which we define as the closure of the union of the regions in $\mathscr{V}(S)$ that belong to points in S_i. Because q is closest neighbor of p it must be that p lies in the closure of q's region in $\mathscr{V}(S - S_i)$. But now q generates a region outside $\mathscr{W}_i$ which does not even touch $\mathscr{W}_i$ and it also generates some region inside W_i. It follows that q's Voronoi region in $\mathscr{V}(S - S_i)$ is disconnected. This is a contradiction because regions of a Voronoi diagram are convex and therefore connected. This implies that if $q \notin S_i$ is a closest neighbor of a point $p \in S_i$, then q is Voronoi neighbor of some point in S_i.

The analysis of the algorithm is fairly straightforward. The number of sets T_i a point belongs to is at most its degree in $\mathscr{V}(S)$, that is, the number of its Voronoi neighbors. Now, the sum of degrees, over all points in S, is at most $6n - 12$ because there are at most $3n - 6$ Voronoi neighbor pairs. It follows that constructing Voronoi diagrams and locating points takes only $O(n \log n)$ time total. Constructing the sets T_i takes only linear time once $\mathscr{V}(S)$ is built. We thus arrive at the main result of this section.

Theorem 3.1. *Given a collection of sets with a total of n points in the plane, there is an algorithm that takes* $O(n \log n)$ *time to find for each point a closest neighbor outside its set.*

Observe that the above algorithm works in the algebraic decision tree model because the only difference between the RAM and the algebraic decision tree model is the indirect addressing capability of a RAM, and we never used indirect addressing in our algorithm.

4. A difference between the RAM and the algebraic decision tree

The $\Omega(n \log n)$ lower bound for the integer element uniqueness problem in [16] implies that there are problems whose complexity in the RAM model differs from its complexity in the algebraic decision tree model, integer element uniqueness being one of them. To show this we solve the integer element uniqueness problem in linear time using a technique described in [1, p.71]. We write $M[i]$ for the memory location with address i and assume that the n given numbers $x_1, x_2, \ldots, x_n$, are stored in $M[X+1]$ through $M[X+n]$, where X is the largest of the x_i. For a given index i we define DOUBLE(i) = **true** if $X + 1 \leqslant M[x_i] \leqslant X + n$, $M[x_i] \neq X + i$, and $M[M[x_i]] = x_i$; otherwise, DOUBLE(i) = **false**. That is, DOUBLE(i) is true if and only if $M[x_i]$ points to the range of memory locations that accommodate the integers, it does not point to $M[X+i]$ which stores x_i, but still the memory location it points to stores a value equal to x_i. The algorithm works without memory initialization.

```
$i := 1$;
while $i < n+1$ and DOUBLE($i$) = false do
    set $M[x_i] := X + i$; $i := i + 1$
endwhile;
if $i < n+1$
    then "yes, there are two equal numbers"
    else "no, all $n$ numbers are different"
endif.
```

Remark. There are two unrealistic assumptions that make the above algorithm work in time $O(n)$: one is that a memory location can hold an arbitrarily large integer, the other that there are arbitrarily many memory locations. Together with these assumptions indirect addressing is powerful enough to admit a RAM program for the integer element uniqueness problem that is faster than the lower bound in the algebraic decision tree model.

5. Remarks and open problems

This paper proves an $\Omega(n \log n)$ lower bound in the algebraic decision tree model for computing a closest pair of vertices of a simple polygon in the plane. This matches the upper bound for the problem and thus settles a problem of Lee and Preparata [11]. In Section 4 we point out a difference between the RAM model and the algebraic decision tree model which explains why our lower bound argument does not apply in the RAM model.

As mentioned in Section 2, a furthest pair of vertices of a simple n-gon can be found in time $O(n)$, which is of course tight. However, to determine the complexity of finding a furthest neighbor for each vertex is still an open problem; the known bounds are $\Omega(n)$ and $O(n \log n)$. However, when the polygon is convex there is an $O(n)$ upper bound matching the trivial lower bound, see [2].

Another proximity problem for simple polygons whose time-complexity is open is to find a closest pair so that the two vertices are visible inside the polygon. A related result is the $O(n \cdot \log\log n)$ upper bound due to [3,9,6] for finding a shortest vertex-to-vertex connection between two non-intersecting simple polygons so that the two vertices are visible from each other. Their algorithm can be extended to find a closest visible vertex pair inside a simple n-gon in time $O(n \log n)$.

Acknowledgment

The authors thank the two referees for carefully reading the manuscript and for suggesting the changes that improve the readability of the paper.

References

[1] A.V. Aho, J.E. Hopcroft and J.D. Ullman, *The Design and Analysis of Computer Algorithms* (Addison-Wesley, Reading, MA, 1974).

[2] A. Aggarwal, M. Klawe, S. Moran, P.W. Shor and R. Wilber, Geometric applications of a matrix searching algorithm, *Algorithmica* **2** (1986) 195–208.

[3] A. Aggarwal, S. Moran, P.W. Shor and S. Suri, computing the minimum visible vertex distance between two polygons, in: *Proc. Workshop on Algorithms and Data Structures*, Ottawa (1989) 115–134.

[4] M. Ben-Or, Lower bounds for algebraic decision trees, in: *Proc. 14th Ann. ACM Symp. on Theory of Computing* (1983) 80–86.

[5] J.L. Bentley and M.I. Shamos, Divide-and-conquer in multidimensional space, in: *Proc. 8th Ann. ACM Symp. on Theory of Computing* (1976) 220–230.

[6] L.P. Chew, Constrained Delaunay Triangulations, in: *Proc. 3rd Ann. ACM. Symp. on Computational Geometry* (1987) 215–222.

[7] H. Edelsbrunner, *Algorithms in Combinatorial Geometry* (Springer, Heidelberg, 1987).

[8] R.L. Graham and F.F. Yao, Finding the convex hull of a simple polygon, *J. Algorithms* **4** (1983) 324–331.

[9] D.T. Lee and A.K. Lin, Generalized Delaunay triangulations of planar graphs, *Discrete Comput. Geom.* **1** (1986) 201–216.

[10] D.T. Lee and F.P. Preparata, The all nearest-neighbor problem for convex polygons, *Inform. Process. Lett.* **7** (1978) 189–192.

[11] D.T. Lee and F.P. Preparata, Computational geometry – a survey, *IEEE Trans. Comput.* **33** (1984) 1072–1101.

[12] D. McCallum and D. Avis, A linear algorithm for finding the convex hull of a simple polygon, *Inform. Process. Lett.* **9** (1979) 201–206.

[13] F.P. Preparata and M.I. Shamos, *Computational Geometry – An Introduction* (Springer, New York, 1985).

[14] R. Seidel, A method for proving lower bounds for certain geometry problems, in: G.T. Toussaint, ed., *Computational Geometry* (North-Holland, Amsterdam, 1985) 319–334.

[15] M.I. Shamos, *Computational Geometry*., Ph.D. Thesis, Dept. of Computer Science, Yale University, New Haven, CT, 1978.

[16] A.C.-C. Yao, Lower bounds for algebraic computation trees with integer inputs, in: *Proc. 30th Ann. IEEE Symp. on Foundations of Computer Science*, 1989, to appear.