1984

Computing the Convex Hull of Line Intersections

Mikhail Atallah
Purdue University, mja@cs.purdue.edu

Report Number:
84-499

TR-499
11-8-84

COMPUTING THE CONVEX HULL OF LINE INTERSECTIONS

Mikhail Atallah

Department of Computer Sciences
Purdue University
West Lafayette, Indiana 47907.

Abstract

We give an $O(n \log n)$ time algorithm for computing the convex hull of the $n(n-1)/2$ points determined by the pairwise intersections of n lines in the plane.

1. Introduction

Consider the set S of $n(n-1)/2$ points determined by the pairwise intersections of n lines in the plane. If the input consists of the n lines, then which properties of S can be computed in time $o(n^2)$? Even though we are computing a property of $\Theta(n^2)$ points, these points have a "compact" description, namely the set of n straight lines which define them. In this note we show that the convex hull of the set S can be computed in time $O(n \log n)$. An immediate consequence of our result is that the farthest pair of points in S, the smallest circle enclosing S, and any other property which depends only on the convex hull of S, can be found in time $O(n \log n)$.

We now introduce some conventions and terminology which will be used throughout the paper. If W is a any set of points, then we use $CH(W)$ to denote the convex hull of W. The storage description we use for $CH(W)$ is a list of the points of W that are on the convex hull, in counterclockwise cyclic order. A point $p \in W$ is a *corner* iff it belongs to $CH(W)$ and it does *not* belong to the straight-line segment which joins the point preceeding it on $CH(W)$ to the one succeeding it on $CH(W)$. The result of deleting all the non-corner points from $CH(W)$ is called the *edge-hull* of W and is denoted by $ECH(W)$. If all the points of W are in general position (i.e. if no three of them are aligned) then every point of the convex hull of W is a corner, and therefore in that case $ECH(W)=CH(W)$. However, the points of the set S under consideration are certainly not in general position, and therefore $ECH(S)$ and $CH(S)$ differ.

In order to simplify the presentation, we assume throughout the paper that the n input lines have distinct slopes (i.e. no two are parallel), that none of them is vertical, and that no three of them meet at a point. (Our discussion can easily be modified for the general case.)

The paper is organized as follows. Section 2 gives an $O(n \log n)$ time algorithm

for computing $ECH(S)$, Section 3 sketches how to obtain $CH(S)$ from $ECH(S)$, Section 4 discusses the dual of this problem, and Section 5 concludes.

2. Computing ECH(S)

The input to the algorithm described in this section are the n lines whose pairwise intersections define the set S. The output of the algorithm is the edge-hull of S, $ECH(S)$. Since the algorithm is supposed to run in time $O(n\log n)$, it is obvious that we cannot afford to explicitly generate the set S. Instead, the algorithm generates a set Q of n points which has the property that $ECH(Q)=ECH(S)$. Once we have such a set Q, $ECH(Q)$ can be computed in time $O(n\log n)$ by using any of the known convex hull algorithms.

Algorithm Edge-Hull

Step 1. Sort the n input straight lines by decreasing slope. Let $L_0, \cdots, L_{n-1}$ be the n lines listed by decreasing slope, i.e. if the equation of L_i is $y=a_i x+b_i$ then we have $a_0 > a_1 > \cdots > a_{n-1}$. This Step takes $O(n\log n)$ time.

Step 2. Let point q_i denote the intersection of lines L_i and $L_{i+1 \bmod n}$. Find the set $Q=\{q_0, \cdots, q_{n-1}\}$. This takes $O(n)$ time.

Step 3. Compute $ECH(Q)$ using any of the known $O(n\log n)$ time convex hull algorithms (e.g. [G]). Then output $ECH(Q)$.

End of Algorithm Edge-Hull

It is obvious that the above algorithm runs in $O(n\log n)$ time. Correctness of the algorithm would immediately follow if we could prove that $ECH(Q)=ECH(S)$. To prove this, it suffices to show that if p is a corner point of S then $p \in Q$. So let p be a corner point of S. Let L_i and L_j, $j>i$, be the two lines whose intersection is p. To prove that $p \in Q$, we must show that either $j=i+1$ (i.e. $p=q_i$), or $j=n-1$ and $i=0$

(i.e. $p = q_{n-1}$). We prove this by contradiction: Suppose to the contrary that $p \neq q_i, q_j$. Since $p \neq q_i$, there is at least one line L_k whose slope is between the slopes of L_j and L_i, i.e. $a_i > a_k > a_j$. Let v and w be the points of intersection of L_k with L_i and L_j, respectively (see Figure 1).

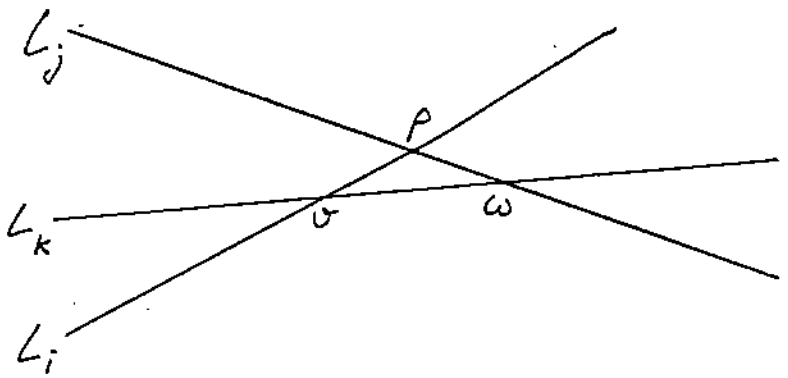

Figure 1

We continue the proof assuming v is to the left of w, as in Figure 1 (the argument when w is to the left of v is entirely symmetrical). Since $p \neq q_{n-1}$, one (or both) of the following is true:

(i) $j \neq n-1$, or

(ii) $i \neq 0$.

If (i) holds then we obtain a contradiction as follows. Line L_{n-1} must cross line L_i somewhere, say at point s. If s is to the right of p then p is on the straight-line segment joining s to v, which contradicts the fact that p is a corner point. If, on the other hand, s is to the left of p, then L_{n-1} intersects L_j at a point t which is to the left of p. This implies that p is on the straight-line segment joining t to w, which contradicts the fact that p is a a corner point. Therefore (i) cannot occur.

The argument for finding a contradiction when (ii) holds is similar to the above argument for (i), except that L_0 plays the role of L_{n-1}. Since either (i) or (ii) leads to a contradiction, it follows that our original assumption (that $p \notin Q$) was wrong.

This completes the proof that every corner point belongs to Q, from which correctness of the algorithm follows. The reader can verify that not every point in Q is a corner point, or even belongs to $CH(S)$ (it is not hard to construct an example where some of the points in Q are in $CH(S)$ but are not corners, while other points of Q are in the interior of the polygon $CH(S)$).

3. Computing CH(S)

The algorithm for computing $CH(S)$ works as follows. We compute $ECH(S)$ in time $O(n \log n)$, using the algorithm of the previous section. Then we mark every edge of $ECH(S)$ which is along one of the input lines as being *special*. This takes $O(n)$ time. Then for every line L_i we do the following. First we find the intersection of L_i with the convex polygon $ECH(S)$. This takes $O(\log n)$ time by using the Chazelle-Dobkin algorithm for intersecting a line with a convex polygon [CD]. Let s_i and t_i be the points of intersection of L_i with $ECH(S)$: If s_i (resp. t_i) is a corner, or the edge of $ECH(S)$ to which s_i (resp. t_i) belongs is special, then add s_i (resp. t_i) to a set H (H is initially empty). Since this is done for every L_i the total cost of creating H is $O(n \log n)$ time, and $|H| \leq 2n$. The set H now contains all the points of S that appear on $CH(S)$, whether they are corners or not, and therefore $CH(H)=CH(S)$. Computing $CH(H)$ $(=CH(S))$ can now be done in time $O(n \log n)$ by using any of the known $O(n \log n)$ time convex-hull algorithms (e.g. [G]).

4. The dual problem

There is a well-known transform π which maps a point into a line and a line into a point, in the following way: A point $p=(a,b)$ maps into the line $\pi(p)$ whose equation is $y=ax+b$, and a line L whose equation is $y=cx+d$ maps into the point $\pi(L)=(-c,d)$. The transform π preserves the *'above'* relationship between points and lines. In other words a point p above a line L maps into the line $\pi(p)$ which is above the point $\pi(L)$ (the same is true if the word 'above' is replaced by 'below' in the previous sentence). We refer the reader to [B] for a proof of the above-mentioned property. The line $\pi(p)$ is called the *dual* of point p, and the point $\pi(L)$ is called the dual of line L. We use $\pi(S)$ to denote the set of lines that are duals of the points in S.

Before proceeding, we review the definitions of the lower and upper envelopes of a set of lines. The *lower envelope* of m lines whose equations are $f_i(x)=\alpha_i x+\beta_i$, $1\leq i\leq m$, is the piecewise linear function $l(x)=\min_{1\leq i\leq m} f_i(x)$. The *upper envelope* is defined by replacing 'min' by 'max' in the definition of $l(x)$.

Let the *lower hull* (resp. *upper hull*) of a set of points denote the lower (resp. upper) portion of their convex hull. A line L which contains an edge of the lower hull of S has the property that no point of S is below it. The dual of the previous statement is that the point $\pi(L)$ has the property that no line of $\pi(S)$ is below it, i.e. $\pi(L)$ is on the lower envelope of $\pi(S)$. Actually, the duals of the lines that contain edges of the lower hull of S are the break points of the lower envelope of $\pi(S)$. A similar correspondence exists between the upper hull of S and the upper envelope of the lines in $\pi(S)$.

Given this duality, it is clear that the problem of computing the convex hull of S is, in the dual world, that of computing the lower and upper envelopes of the lines in $\pi(S)$. Therefore our algorithm can also be used for computing, in time $O(n\log n)$, the upper (or lower) envelope of the $n(n-1)/2$ lines determined by n input points.

5. Conclusion

We gave an $O(n \log n)$ time algorithm for computing the convex hull of the $n(n-1)/2$ points determined by the pairwise intersections of n input lines. Therefore all the properties of these points which depend on their convex hull only can also be computed within the same time bound. It would be interesting to know which other properties of the points can also be computed without enumerating them. More generally, what kinds of questions about these points can be answered easier than for an arbitrary set of $\Theta(n^2)$ points? One would expect the fact that these points are the intersections of n lines to be useful, and yet it is often not clear how to exploit this fact. For example, can the points be sorted according to (say) their x-coordinates in $O(n^2)$ time?

Acknowledgement. A useful conversation with Joseph O'Rourke is gratefully acknowledged.

References

[B] K. Q. Brown, 'Geometric transforms for fast geometric algorithms,' Report CMU-CS-80-101, Department of Computer Science, Carnegie-Mellon University, 1980.

[CD] B. Chazelle and D. P. Dobkin, 'Detection is easier than computation,' *Proc. of 12th Annual ACM Symposium on Theory of Computing*, pp. 146-153, 1980.

[G] R. L. Graham, 'An efficient algorithm for determining the convex hull of a finite planar set,' *Information Processing Letters*, vol. 1, pp. 132-133, 1972.

[S] M. I. Shamos, 'Computational geometry', Ph.D. thesis, Dept. of Computer Science, Yale U., New Haven, Conn., 1977.