NOTES ON MAINTENANCE OF CONFIGURATIONS IN THE PLANE

Mark H. Overmars and Jan van Leeuwen

RUU-CS-80-5

September 1980

Rijksuniversiteit Utrecht

Vakgroep informatica

Princetonplein 5
Postbus 80.002
3508 TA Utrecht
Telefoon 030–531454
The Netherlands

NOTES ON MAINTENANCE OF CONFIGURATIONS IN THE PLANE

Mark H. Overmars and Jan van Leeuwen

Technical Report RUU-CS-80-5

September 1980

Department of Computer Science

University of Utrecht

P.O. Box 80.002, 3508 TA Utrecht

the Netherlands

NOTES ON MAINTENANCE OF CONFIGURATIONS IN THE PLANE

Mark H. Overmars* and Jan van Leeuwen

Department of Computer Science, University of Utrecht
P.O. Box 80.002, 3508 TA Utrecht, the Netherlands

Abstract. We give a method to find the convex hull of two separated convex polygons in O(log n) time, where n is the joint number of points of the convex polygons. We also give a method to find the intersection of two convex arcs, with bounding edges separated by a known direction in O(log n) time. In this way we improve two crucial bounds in a paper of Overmars and van Leeuwen [2], which have a number of important applications. We list the new bounds that are obtained by using our new methods. It follows, for example, that we can maintain the convex hull of a set of points in $O(\log^2 n)$ steps per insertion or deletion.

1. Introduction.

In a recent paper, Overmars and van Leeuwen [2] describe a method for maintaining configurations in the plane. The configurations they consider are the convex hull of a set of points, the common intersection of a set of halfspaces and the contour of maximal elements of a pointset. The dynamization is obtained by applying one master technique, based on an appropriate notion of decomposability for each of the problems. The decomposability consists in each case of a way of determining the configuration for the full set from the same configurations for two, in some way separated, subsets.

For the maintenance of the convex hull of a set of points, Overmars and van Leeuwen [2] first split the convex hull in a left part (the lc-hull) and a right part (the rc-hull). Assume the points p_1 to p_n of the set are ordered by y-coordinate. The notion of decomposability for lc-hulls (rc-hulls) is the following: Given the lc-hull (rc-hull) of $p_1, \ldots, p_i$ and $p_{i+1}, \ldots, p_n$ one can find the lc-hull (rc-hull) of $p_1, \ldots, p_n$ in "little" time. In section 2 we show that the bound of $O(\log^2 n)$ steps given in [2] for this task can be improved to O(log n). Several consequences of this improvement are listed.

---

* This author was supported by the Netherlands Organization for the Advancement of Pure Research (Z.W.O.).

To maintain the common intersection of a set of halfspace, Overmars and van Leeuwen [2] distinguish between the halfspaces facing left and the halfspaces facing right and maintain these subsets separately. Given the intersections of these two sets (which are open convex polygons), the common intersection can be found in $O(\log^2 n)$ steps. Assume the halfspaces h_1 to h_n are ordered with respect to the angle of the bordering line. To maintain the set of halfspaces facing left (and likewise, the other subset) the following decomposability property is exploited: Given the intersection of $h_1, \ldots, h_i$ and $h_{i+1}, \ldots, h_n$ we can find the intersection of $h_1, \ldots, h_n$ in "little" time. The intersection of a set of halfspaces is an open convex arc with its last edges extended to infinity. Finding the intersection of the two "separated" arcs consists of finding the one point of intersection. In section 3 we show how the original bound of $O(\log^2 n)$ steps reported in [2] can be improved to $O(\log n)$ steps. We list the consequences of this improvement to the applications given in [2]. For the maintenance of the contour of maximal elements of a set of points an efficient algorithm was already given in [2] and no further improvement can be obtained following the same technique. We assume familiarity with [2] in the remaining sections.

2. Convex hulls.

The problem we consider in this section is the following: Given a set of points $p_1, \ldots, p_n$ ordered by y-coordinate and the lc-hulls of $p_1, \ldots, p_i$ and of $p_{i+1}, \ldots, p_n$, for some $0 < i < n$, in appropriate data structures, find the lc-hull of $p_1, \ldots, p_n$. Let the lc-hull of $p_1, \ldots, p_i$ be A and of $p_{i+1}, \ldots, p_n$ be C (see figure 1). The lc-hull of $p_1, \ldots, p_n$ consists of a front part of A, a "bridge" B and a last part of C. B is the one common tangent of A and C.

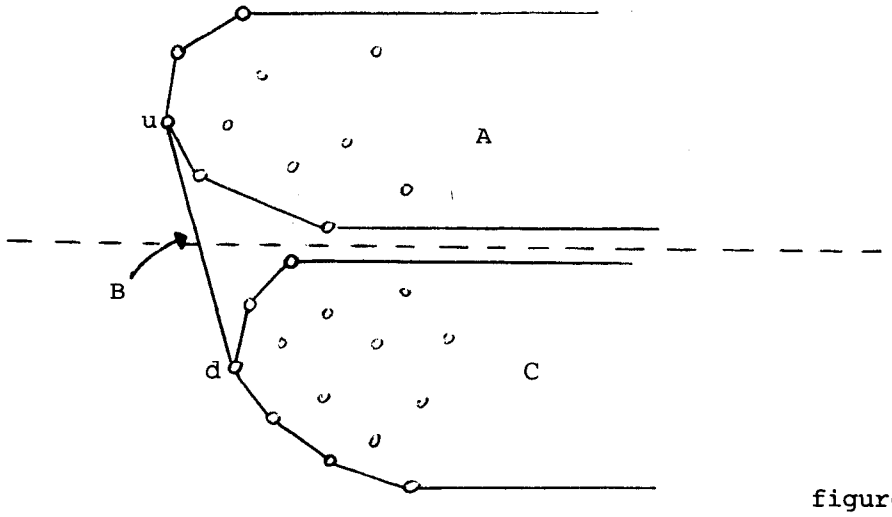


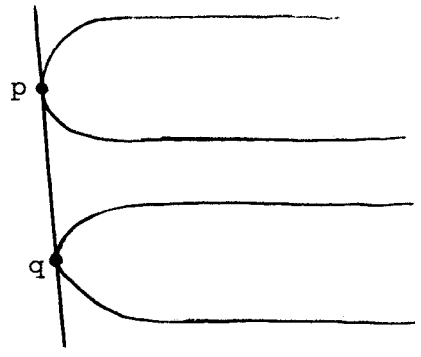
figure 1

Theorem 2.1. Given the lc-hull of $p_1, \ldots, p_i$ and of $p_{i+1}, \ldots, p_n$, the one common tangent B can be found in O(log n) steps.

Proof

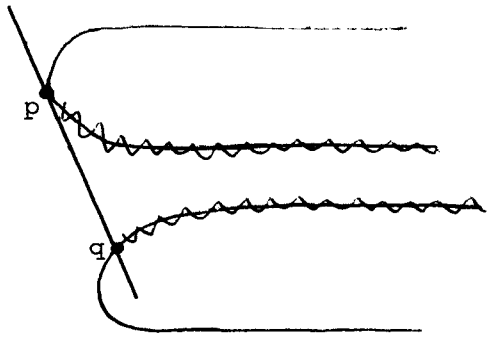
lc-hulls will be presented as concatenable queues. Given a point p on A and a point q on C we would like to have a criterion that enables us to throw the parts before or after p and q away. Once we have such a criterion, repeatedly choosing p and q in the middle of the remaining parts of A and C enables us to find u and d, hence B, in O(log n). Let us look at the way $\overline{pq}$ intersects A and C. The following cases can occur.

case a:



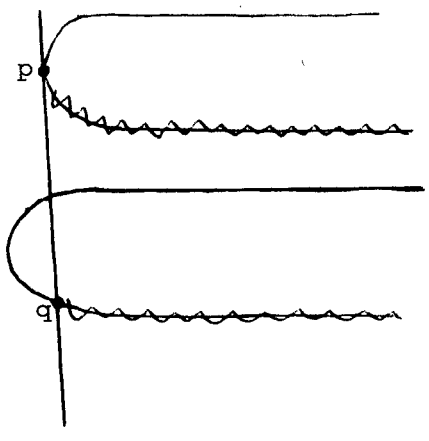
In this case p=u and q=d and we are done.

case b:



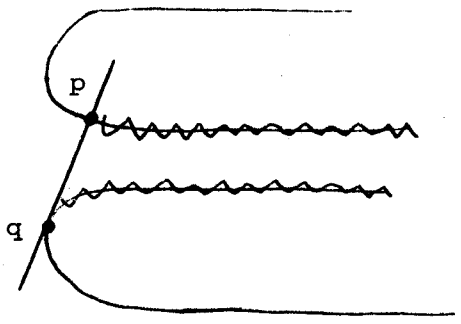
In this case the part of C before q can be deleted, and also the part of A after p.

case c:



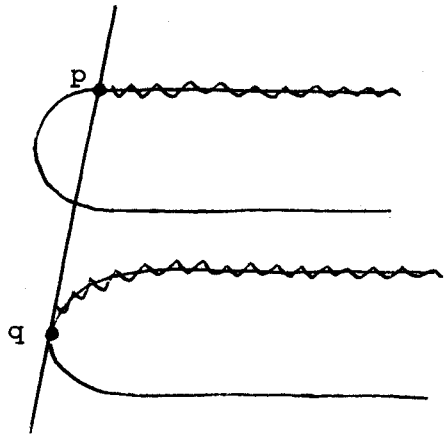
In this case the part of C following q can be deleted, and also the part of A following p.

case d:



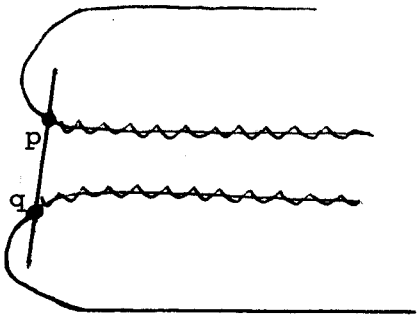
Similar to case b.

case e:



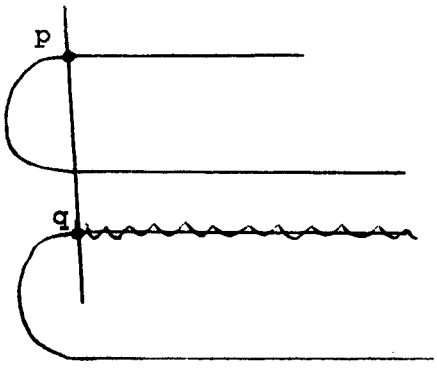
Similar to case c.

case f:



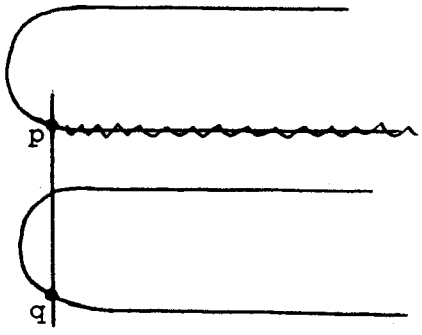
Again a part of both A and C can be deleted.

case g:

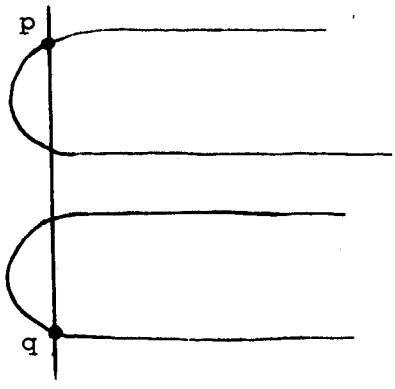


In this case only a part of C can be deleted.

case h: Similar to case g.



case i: This time we cannot say immediately what part of A or C can be deleted. This follows from the fact that when e.g. u lies is the bottom part of A, d may lie on any side of q.



Let us consider case i more carefully. (see figure 2.) Let m be the dividing line of the pointset. Let l_p be a tangent through p, let l_q be a tangent

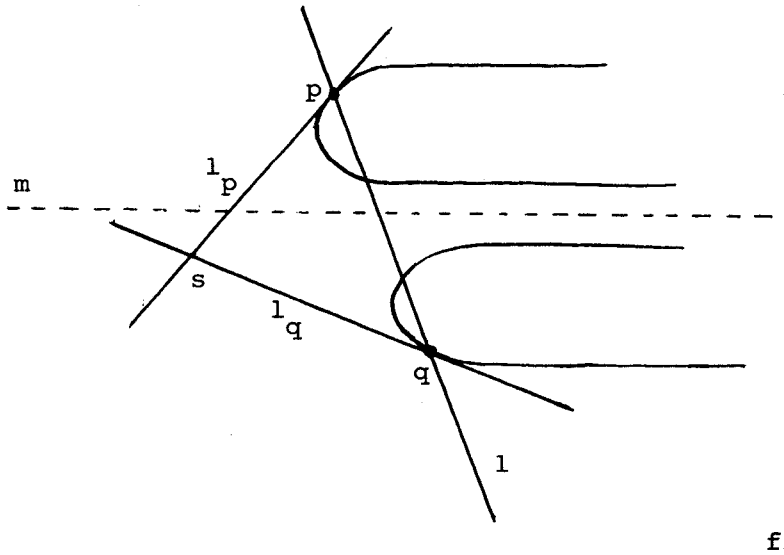
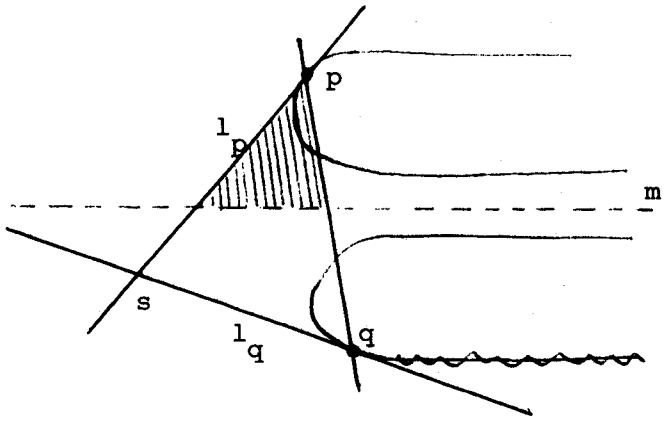


figure 2

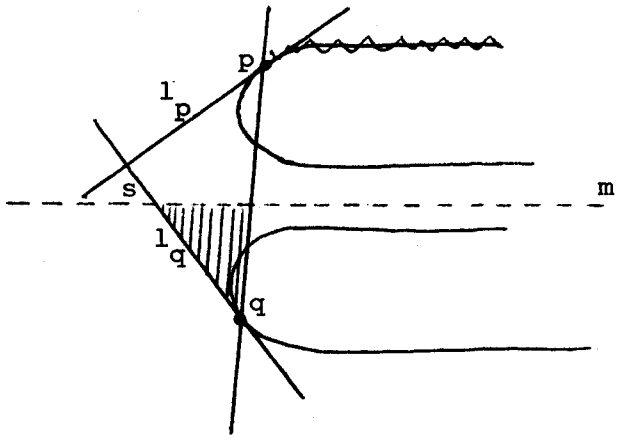
through q and let l be the line through p and q. l_p and l_q intersect at some point s left of l. The following two cases can occur.

case i1: s lies below (or on) m.



Clearly, u can only lie in the shaded area. It follows that u lies above l_q. Hence we can delete the part of C following q.

case i2: s lies above m.



In this case the argument is completely similar and we can delete the part of A before p.

It follows that in all cases a-i we either reach a decision or can delete half of A and/or C. Because all cases can be distinguished in constant time, the total time needed to find B is bounded by O(log n). After B has been found, the operations are undone to restore the concatenable queues representing A and C.

□

Hence we have proved that the bound $J_1(n)$ of [2] indeed is O(log n) instead of $O(\log^2 n)$. In [2] many applications were given that can now be improved. (The numbers in brackets give the theorem number in [2].)

Theorem 2.2.(4.6.) The convex hull of a set of n points in the plane can be maintained at a cost of $O(\log^2 n)$ per insertion and deletion.

Theorem 2.3.(5.1.) One can peel a set of n points in the plane in only $O(n \log^2 n)$ steps.

Theorem 2.4.(5.2.) One can determine the joint convex layers of a set of n points in the plane (hence Barnett's c-order group) in only $O(n \log^2 n)$ steps.

Theorem 2.5.(5.3.) Given n points in the plane, one can determine a connecting spiral in only $O(n \log^2 n)$ steps.

Theorem 2.6.(5.4.) One can maintain a set F of n points in the plane at a cost of $O(\log^2 n)$ time per insertion or deletion, such that queries of the form "does x belong to the interior of the current convex hull of F" can be answered in $O(\log n)$ steps.

Using a result of Chazelle and Dobkin [1] we also have

Theorem 2.7.(5.6.) One can maintain two sets A and B of points in the plane such that insertions and deletions take $O(\log^2 n)$ (where n is the current size of the set on which they operate) and, whenever needed, separability can be decided in only $O(\log n)$ time.

3. Intersection of halfspaces.

The second problem we consider is the following: Given a set of halfspaces ordered by angle and facing leftwards and the common intersections of $h_1, \ldots, h_i$ and of $h_{i+1}, \ldots, h_n$, find the common intersection of h_1 to h_n. The intersection of a set of halfspaces can best be viewed as a convex arc, with the last edges extended to infinity, that bounds the intersection. We call the arc the boundary of the intersection. Let the boundary of $h_1, \ldots, h_i$ be A and the boundary of $h_{i+1}, \ldots, h_n$ be C. The boundary of the total set consists of a front part of A and a tail part of C (see figure 3). Hence all we have to do is find the point of intersection s.

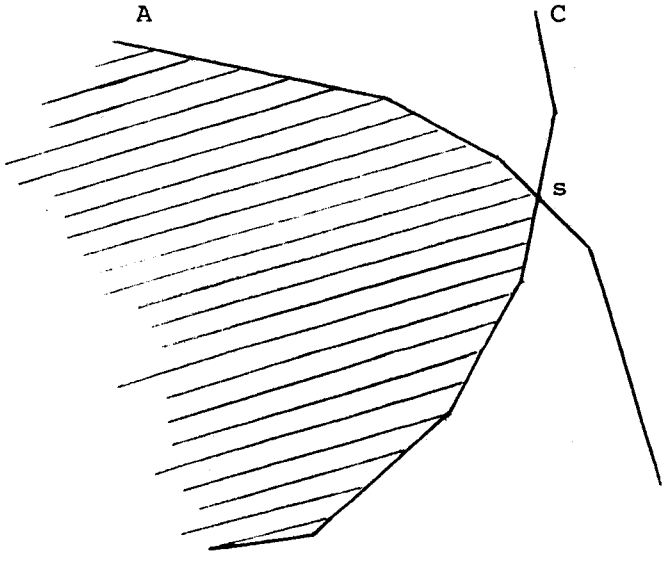


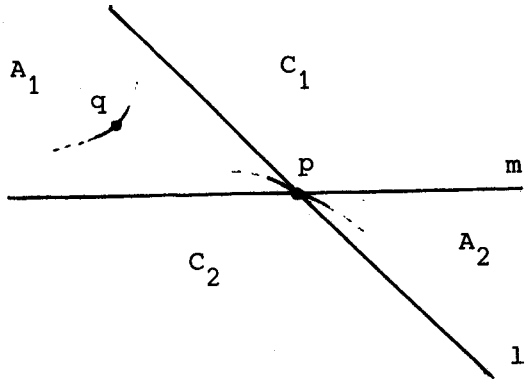
figure 3

Theorem 3.1. Given the boundary of $h_1, \ldots, h_i$ and of $h_{i+1}, \ldots, h_n$, we can find the intersecting point s in O(log n).

Proof

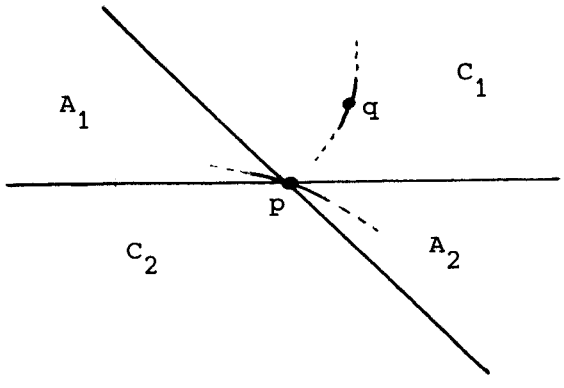
We will follow a similar approach as in theorem 2.1. The boundaries A and C are represented as concatenable queues and we want that, given a point p on an edge of A and a point q on an edge of C, we can delete half of A or C. Let l be a line through p with angle between h_i and h_{i+1} (thus "separating" A and C by angle) and let m be a horizontal line through p. l and m divide the plane in four regions A_1, C_1 A_2 and C_2 as in the figures below. Note that A runs only through A_1 and A_2. The following cases can occur:

case a: q lies in A_1.



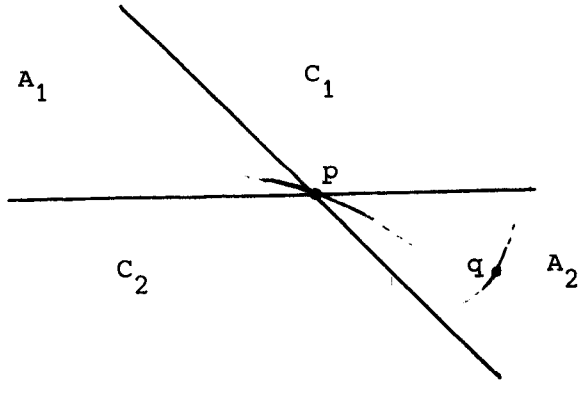
In this case C can never run through A_2. Hence we can delete the part of A below p.

case b: q lies in C_1.



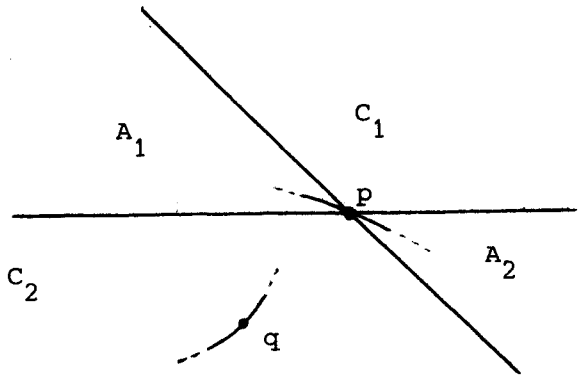
In this case the part of C above q must lie in C_1 and therefore cannot intersect A. Hence we can delete it.

case c: q lies in A_2.



In this case C cannot run through A_1 and we can delete the part of A above p.

case d: q lies in C_2



In this case the part of C below q must lie in C_2 and cannot intersect A. Hence we can delete it.

Hence in each case we can delete half of A or half of C. When A or C is reduced to one or two edges, we can determine the intersection directly (which takes at most O(log n)). Because the cases a-d can be distinguished in constant time, it follows that we can find s in O(log n). Afterwards all operations are undone, to restore the representations of A and C.

□

It follows that we have improved the bound $J_2(n)$ of [2] to be O(log n). There are a number of applications of this result in [2], which can consequently be improved.

Theorem 3.2.(7.2.) The common intersection of a set of halfspaces in the plane can be maintained at a cost of $O(\log^2 n)$ per insertion or deletion, where n is the current number of halfspaces in the set.

Theorem 3.3.(7.3.) One can dynamically maintain the common intersection of a set of halfspaces in the plane such that insertions and deletions take $O(\log^2 n)$ and queries of the form "does x belong to the current common intersection" can be answered in only O(log n), where n is the number of halfspaces in the set.

Theorem 3.4.(7.4.) One can dynamically maintain the kernel of a simple n-gon at a cost of only $O(\log^2 n)$ steps per transaction, assuming that transactions merely involve the insertion and/or deletion of some edges that keep the polygon simple.

Theorem 3.5.(7.5.) One can dynamically maintain the feasible region of a

linear program in 2 variables at a cost of $O(\log^2 n)$ for each inequality added or deleted.

4. References.

[1] Chazelle, B. and D.B. Dobkin, Detection is easier than computation, Proc. 12th Annual ACM Symp. on Theory of Computing, Los Angeles, 1980, pp. 146-153.

[2] Overmars, M.H. and J. van Leeuwen, Maintenance of configurations in the plane, Techn. Rep. RUU-CS-79-9, Dept. of Computer Science, University of Utrecht, 1979/1980.