アルゴリズム 47－2
（1995．9．21）

多角形包含を決定する問題について

Antonio Hernández Barrera

広島大学理学部数学科

ここでは私は移動できる全領域を構成しないで、多角形Pを別の固定された多角形Qの中へ移動できるかどうかを決定する問題を考える。私は一般的な場合　すなわち、PとQが任意の単純多角形の場合）とPとQがレクトリニアルの場合についてアルゴリズムを与える。単純多角形のときは$O(n^2m^2)$時間$O(nm)$記憶量でレクトリニアルのときは$O(nm\ log\ nm)$時間、$O(nm)$記憶量である。ここで、nはQの頂点の数で、mはPの頂点の数である。

The Polygon Containment Decision Problem

Antonio Hernández Barrera

Department of Mathematics, Faculty of Science, Hiroshima University
e-mai:r5m02@math.sci.hiroshima-u.ac.jp

We investigate here the problem of deciding whether a polygon P can be translated to fit inside another polygon Q without constructing the whole feasible region. We present algorithms for the general case, i.e. P and Q being any simple polygons, and for the rectilinear one. We obtain $O(n^2m^2)$ and $O(nm\ log\ nm)$ time complexities, respectively, where n is the number of vertices of Q and m is the number of vertices of P. Both solutions use $O(nm)$ space.

1 Introduction

The polygon containment problem is the problem of deciding whether a given polygon P can be moved to fit inside another given polygon Q. For more than ten years this problem has been studied and several results have been given by [2], [3], [4] and [9]. There, algorithms for different cases of polygon shapes and motions have been considered but all of them are concerned with finding the whole feasible region of P inside Q.

Some results concerning the polygon containment problem without constructing the whole boundary of the feasible placements appear in [6], where in the case under translation, a family of decision algorithms for monotone polygons is presented. Specifically, the authors give algorithms for problems in which (1) both polygons are rectilinearly convex, (2) both polygons are rectilinearly 2-concave and (3) both polygons are monotone. Using a plane-sweep-like approach, their algorithms terminate when the first feasible placement is found, if one exists.

In this paper we extend the results in [6] to the case where P and Q are any simple polygons and to the case where P and Q are rectilinear polygons. Indeed, our results are still valid even if both P and Q have holes and consist of several disjoint polygons. We present algorithms with a worst-case time complexity of $O(n^2m^2)$ for the simple case and of $O(nm\ log\ nm)$ for the rectilinear one, where m and n are the numbers of the vertices of P and Q, respectively. The space is $O(nm)$.

In the rectilinear case, the complexity of our algorithm is lower than the time necessary to construct the feasible region, since this one may have $O(n^2m^2)$ vertices.

2 Preliminaries

Suppose that P and Q are polygons, P can translate while Q is fixed in the plane. A position of P in the plane is uniquely determined by the position of a specific point of P. This point is called the *reference point*. A *valid* or *feasible placement* is a placement of the polygon P, that is, the reference point, such that P is contained in Q. A *feasible region* is the set of all feasible placements of P which is denoted here by $C(P,Q)$. In general, $C(P,Q)$ may consist of a finite number of polygons, line segments and points. We consider that the boundary of P can "touch" the boundary of Q. Hence, $C(P,Q)$ is a closed set since the points in the boundary of $C(P,Q)$ correspond to positions of the reference point where the boundaries of both polygons share at least one point.

We denote $CH(P)$ the convex hull of polygon P and $O(P,Q)$ the set of locations that make P to intersect Q. For convenience, we consider $O(P,Q)$ to be an open set. $A \cap B$, $A \cup B$ and $A - B$ have the usual interpretations: intersection, union and the set of elements which belong to A but not to B.

A polygon P is *simple* if its edges are non-intersecting. P is *rectilinear* if it is simple and the edges of its boundary are either ver-

tical or horizontal. A *hole* of the union of a set of (open) regions in the plane, is a connected bounded component of the complement of the union of the regions.

3 The general case

Suppose we are given P and Q, any two simple polygons with m and n edges, respectively. The problem of computing the feasible region of P inside Q has been attacked in [2] where a solution is given to the case under translation using $O(n^2m^2log\ mn)$ time. This solution is nearly optimal as the feasible region may have $\Omega(n^2m^2)$ vertices in the worst case ([3]). The method can be generalized in case rotations are also allowed to obtain a $O(n^3m^3log\ mn)$ time complexity algorithm. There is also another solution for the case under translation:

Proposition 3.1 *Suppose that the concavities of polygon Q $(CH(Q) - Q)$ are decomposed into convex parts $Q_1, ..., Q_k$ and that polygon P is also decomposed into convex parts $P_1, ..., P_l$. The set of valid placements is given by $C(CH(P), CH(Q)) - \bigcup_{ij} O(P_i, Q_j)$.*

Based on Proposition 3.1, Avnaim and Boissonnat ([1]) developed an algorithm for computing the feasible placements with time complexity $O(c_qc_p(c_pn + c_qm)\ log\ (c_pn + c_qm))$. Here, c_p denotes the number of concave vertices of P plus one and c_q denotes the number of convex vertices of Q which are not vertices of the convex hull of Q plus the number of edges of the convex hull of Q which are not edges of Q. In the worst case, $c_q = O(n)$ and $c_p = O(m)$, so the time complexity becomes again $O(n^2m^2log\ nm)$.

In this section, we show how to solve the detection problem using $O(n^2m^2)$ time and $O(nm)$ space.

Let k' be the number of edges of $C(CH(P), CH(Q))$. Consider the supporting line of edge e_i for each e_i in the boundary of $C(CH(P), CH(Q))$. Let h_i be the halfplane which the interior of $C(CH(P), CH(Q))$ is not in.

Theorem 3.2 *P can be placed inside Q iff $\cup_{j=1}^{k} \cup_{i=1}^{l} O(P_i, Q_j) \bigcup \cup_{i=1}^{k'} h_i$ has a hole.*

The algorithm that determines whether P fits inside Q proceeds in four steps:

Algorithm S

Step 1 Compute $CH(Q)$, $CH(P)$ and $C(CH(P), CH(Q))$;

Step 2 Obtain a decomposition of $CH(Q) - Q$ into convex parts $Q_1, ..., Q_k$ and a decomposition of P into convex parts $P_1, ..., P_l$;

Step 3 Compute $O(P_i, Q_j)$ $\forall i, j 1 \leq i \leq l, 1 \leq j \leq k$;

Step 4 Determine whether $\cup_{j=1}^{k} \cup_{i=1}^{l} O(P_i, Q_j) \bigcup \cup_{i=1}^{k'} h_i$ has a hole, where h_i and k' are as above;

The correctness of Algorithm S follows immediately. Let us analyze its performance time. Step 1 takes $O(n + m)$ time because the convex hull of a polygon can be

computed in linear time ([14]) and the set of translations that fit a convex m-gon inside another convex n-gon is a convex polygon, with at most n edges, that can be determined in $O(n+m)$ ([4]). To triangulate a simple polygon is to partition it into triangles without adding new vertices. It is possible to obtain such a decomposition, into a number of triangles which is less than twice the number of edges of the polygon, using $O(u\ log\ u)$ time, where u is the number of edges of the polygon ([11]). On the other hand, notice that $CH(Q)-Q$ is a set of disjoint polygons, with at most n edges, that can be computed in linear time. Therefore, Step 2 can be performed in $O(n\ log\ n + m\ log\ m)$ yielding $k < 2n$ and $l < 2m$ convex parts. It is easy to see that $O(P_i, Q_j)$ is a convex polygon with at most six vertices which can be computed in constant time so Step 3 takes $O(mn)$ time.

A solution to the problem of determining whether the union of the set of convex polygons of Step 4 has a hole can be based on computing the arrangement of the supporting lines of the polygon edges. Once the arrangement has been obtained we can mark the cells (vertices, edges and regions) contained in some polygon using constant time per cell. There is a hole if and only if there is at least one unmarked cell. The arrangement of N lines in the plane can be computed in $O(N^2)$ time ([8]). In our case, the set of lines would have size $O(kl+k')$ which is $O(nm)$. Step 4 then takes $O(n^2m^2)$.

In summary, the detection problem can be solved in $O(n^2m^2)$. The storage is also $O(n^2m^2)$ since the arrangement will be constructed entirely. Indeed, using this approach we can compute all the unmarked edges what allows us to have a description of the whole feasible region. In this way, we obtain an algorithm with better performance time than the ones in [1] and [2].

However, our goal is to solve the detection problem and we demonstrate now that there is a way to solve it without having to construct the whole arrangement. The solution uses the topological sweep described in [7].

In the topological sweep the arrangement is swept by a line that is not necessarily straight. In the process each cell is visited once and the elementary steps (an *elementary step* is performed when the topological line sweeps past a vertex) along a given line are performed from left to right. The scheme uses quadratic time but only linear space is necessary if the cells are examined as they are built and discarded immediately afterwards.

Let us associate to each line l of the arrangement a counter $c(l)$ to store the number of polygons currently covering l. Clearly, $c(l)$ changes only if a vertex on l is reached. Thus, $c(l)=0$ means that the points on l immediately to the right of a vertex where an elementary step has been performed at are not inside any polygon, $c(l)=1$ means that those points are interior to one polygon and so on. For each line l $c(l)$ can be updated during the sweep like follows.

Initially let $c(l)=1$, for l is covered, at $x=-\infty$, only by the complement of $C(CH(P), CH(Q))$. Suppose that an ele-

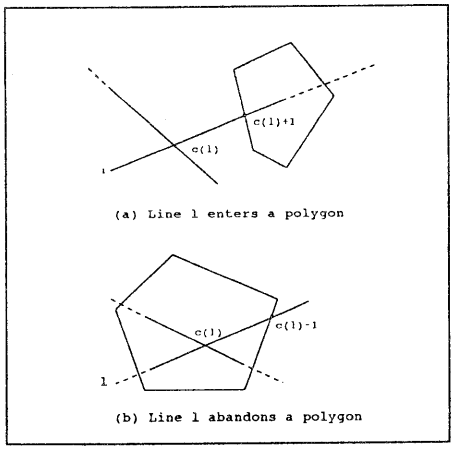

Figure 1: Updating c(l)

mentary step is going to be performed at a vertex on l. The value of $c(l)$ is modified in this way: $c(l) = c(l) + 1$ if line l enters a polygon or c(l)=c(l)-1 if line l abandons a polygon. See Figure 1. Notice that determining whether a line enters or abandons a polygon at a vertex can be done in $O(1)$.

It is clear that an edge of the arrangement lying on l, is part of a hole if and only if $c(l) = 0$ between the two vertices of the edge. Thus, we can determine whether polygon P fits inside Q by topologically sweeping the arrangement and computing the counter values. Observe that the algorithm can stop whenever $c(l) = 0$ for some line l without scanning the rest of the arrangement since the answer for the detection problem is already known. In summary,

Theorem 3.3 *Let P and Q be two simple polygons with m and n edges, respectively. Determining whether P can be translated to fit inside Q can be done, in the worst case, in $O(n^2m^2)$ time and $O(nm)$ space.*

Algorithm S can be applied even if both P and Q have holes (P must fit inside Q but outside Q's holes) and consists of several disjoint polygons, since Proposition 3.1 and Theorem 3.2 still hold.

4 The rectilinear case

Suppose that P and Q are two rectilinear polygons with m and n edges, respectively. There is an $O(n^2m^2)$ time complexity algorithm in [1] for computing $C(P, Q)$, which is worst-case optimal since, as we have already said, the feasible region may have $\Omega(n^2m^2)$ vertices. There, the authors raised the question of whether the complexity of their algorithm could be reduced in case we only want to decide containment. In the following, we prove that the answer to that question is affirmative.

First, we recall some results which we shall use in the sequel. Given a set S of possible overlapping intervals on a line, consider the problem of maintaining efficiently, under insertions and deletions in S, the union of the intervals in S. This problem has been considered first in [13] and later in [5], where an improved solution is obtained by storing the intervals in a red-black tree ([10]) suitably augmented with additional pointers to allow fast reporting and update. From [5] we take this proposition:

Proposition 4.1 *Given a set S of intervals on a line, the ordered list of intervals in their union can be reported in $\theta(k)$ time and insertion/deletion of an interval can be done in $O(\log n)$ time in the worst case, where k is the number of intervals in the union and n is the number of intervals currently in S. The scheme uses $\theta(n)$ space.*

As in Algorithm S, we base our solution of the rectilinear case on Proposition 3.1 and Theorem 3.2 but, to take advantage of the rectilinear nature of the input, we design a different approach. The idea is to make a sweep of the polygonal regions $\bigcup_{ij} O(P_i, Q_j)$ and $C(CH(P), CH(Q))$ trying to find a portion of the latter not covered by the former. $CH(Q)$ will not stand this time for the convex hull of the polygon Q. To keep working with rectilinear polygons, instead of the convex hull we will compute the *rectilinearly convex hull* which can be defined as a rectilinearly convex polygon that contains Q and has a minimal area [12].

Algorithm R

Step 1 Compute $C(CH(P), CH(Q))$ and place rectangles along its boundary as in Figure 2. Let them be $h_1, h_2, \ldots, h_{k'}$

Step 2 Decompose $CH(Q) - Q$ into rectangles; say $Q_1, \ldots, Q_k$

Step 3 Decompose P into rectangles; say $P_1, P_2, \ldots, P_l$

Step 4 Compute $O(P_i, Q_j)$ $\forall i, j, 1 \leq i \leq l, 1 \leq j \leq k$

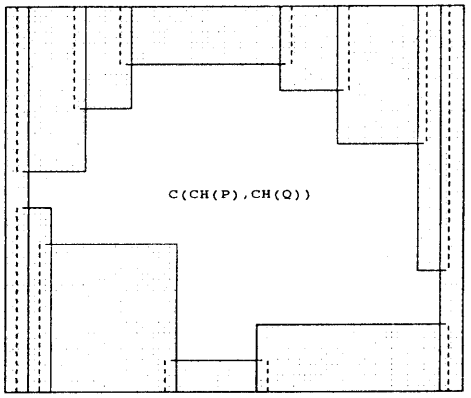

Figure 2: The rectangles h_i are shown shaded

Step 5 Make a sweep of the abscissae of the left and right edges of the $O(P_i, Q_j)$ and h rectangles, following the scheme in Proposition 4.1, i.e. inserting every left edge and deleting every right edge when the corresponding abcissae is reached. Check in each update if a hole has appeared. If so, report that the search finished satisfactorily and stop.

Step 6 Report that there is not room inside Q for P (so far, all the rectangles have been swept and no hole appeared) and stop.

Computing $CH(P)$ and $CH(Q)$ is done in $O(m)$ and $O(n)$, respectively ([12]). $C(CH(P), CH(Q))$ is a rectilinearly convex polygonal region, with at most mn bounding edges, that is computed in $O(nm \log m)$

([3]). Thus, Step 1 takes $O(nm\ log\ m)$. $CH(Q) - Q$ is, as in Algorithm S, determined in linear time. A simple partitioning of a rectilinear N-gon into rectangles is to draw vertical lines at each concave vertex. The number of rectangles obtained in this way is upperly bounded by $N/2$. Such a partitioning can be computed in $O(N\ log\ N)$ time by means of the plane sweep method supported by a balanced tree. Using this idea, $O(n)$ and $O(m)$ rectangles can be computed in Step 2 and Step 3 to yield $O(n\ log\ n)$ and $O(m\ log\ m)$, respectively. This time $O(P_i, Q_j)$ is a rectangle, therefore, Step 4 takes $O(mn)$ time.

To make the plane sweep as described in Step 5, we can sort the left and right edges of the $O(P_i, Q_j)$ and h rectangles by abcissae in $O(mn\ log\ mn)$. There are such $O(mn)$ edges, so inserting/deleting them takes, due to Proposition 4.1, $O(mn\ log\ mn)$ in total. Finally, to detect whether a hole has appeared, we just need to report the list of intervals in the union after an update has been performed. Clearly, there is a hole if and only if there is more than one interval in the union. Also, by Proposition 4.1 the list of intervals can be reported in $O(k)$ where k is the number of the intervals. Because the algorithm will keep running only if there is one interval, the detection step will take $O(1)$ always except possibly once, i.e. when a hole appears. At that moment k could be $O(mn)$ in the worst case since there are $O(mn)$ rectangles and, at any fixed position of the sweep line, each of them can only contribute, at most, one interval to the union.

We have proven the following theorem then:

Theorem 4.2 *Suppose we are given P and Q, two rectilinear polygons with m and n edges, respectively. Determining whether P can be translated to fit inside Q can be done in $O(mn\ log\ mn)$ time and $O(nm)$ space.*

As in Section 3, Theorem 4.2 still holds even if both P and Q have holes and consist of several disjoint polygons.

5 Discussion

We have given here algorithms for detecting whether a given polygon P can be translated to fit inside another given polygon Q. Specifically, we have solved the general (any simple polygons P and Q) and the rectilinear case. In the rectilinear case, the time complexity is lower than the time necessary to compute the whole feasible region. To the best of our knowledge there are not such kinds of results in the case of simpler shaped polygons, namely convex, rectilinearly convex, star polygons and so on. The ideas presented in this paper do not seem to work on these cases. Indeed, for the polygon containment problem, the question of whether the detection problem is "easier" than the computation one is open in general. We have shown in this paper that the answer for the rectilinear case under translation is "yes".

References

[1] F. Avnaim and J.D. Boissonnat. Simultaneous containment of several poly-

gons. In *3rd ACM Symposium on Computational Geometry*, pp. 242–250, 1987.

[2] F. Avnaim and J.D. Boissonnat. Polygon placement under translation and rotation. In *STACS 88, Lecture Notes in Computer Science 294*, pp. 322–333, 1988.

[3] B.S. Baker, S.J. Fortune, and S.R. Mahaney. Polygon containment under translation. *Journal of Algorithms*, No. 7, pp. 532–548, 1986.

[4] B. Chazelle. *Advances in Computer Research*, Vol. 1, pp. 1–33. JAI Press Inc., 1983.

[5] S.W. Cheng and R. Janardan. Efficient maintenance of the union of intervals on a line, with applications. *Journal of Algorithms*, No. 12, pp. 57–74, 1991.

[6] J.S. Chiu and J.S. Wang. Monotone polygon containment problems under translation. *Journal of Information Processing*, Vol. 13, No. 4, pp. 486–493, 1990.

[7] H. Edelsbrunner and L.J. Guibas. Topologically sweeping an arrangement. *Journal of Computer and System Sciences*, Vol. 38, pp. 165–194, 1989.

[8] H. Edelsbrunner, J. O'Rourke, and R. Seidel. Constructing arrangements of lines and hyperplanes with applications. *SIAM Journal of Computing*, Vol. 15, No. 2, pp. 341–363, May 1986.

[9] S.J. Fortune. A fast algorithm for polygon containment by translation. In *12-th ICALP, Lecture Notes in Computer Science 194*, pp. 189–198, 1985.

[10] L. Guibas and R. Sedgewick. A dichromatic framework for balanced trees. In *19th Annual IEEE Symposium on Foundations of Computer Science*, pp. 8–21, 1978.

[11] S. Hertel and K. Mehlhorn. Fast triangulation of the plane with respect to simple polygons. *Information and Control*, pp. 52–76, 1985.

[12] T.M. Nicholl, D.T. Lecand Y.Z. Liao, and C.K. Wong. On the X-Y convex hull of a set of X-Y polygons. *Bit*, No. 23, pp. 456–471, 1983.

[13] M.H. Overmars. Dynamization of order decomposable set problems. *Journal of Algorithms*, No. 2, pp. 245–260, 1981.

[14] F.P. Preparata and M.I. Shamos. *Computational Geometry, an Introduction*, pp. 160–165. Springer-Verlag, 1985.