数理解析研究所講究録
第872巻 1994年 35-40

A solution for a polygon containment problem using sorting $X+Y$.

Antonio Hernández Barrera
Hiroshima University

Abstract

Here an algorithm is presented which shows that the problem of computing the path that a rectilinear non decreasing polygonal takes as it slides along the edges of another one can be reduced to the problem of sorting $X+Y$. The existence of such an algorithm allows us to say that the polygon containment problem under translation in the case of rectilinearly convex polygons can be also reduced to sort $X+Y$.

1 Introduction

The problem of deciding if a given polygon P, which is allowed to have some kind of motions, can be placed into another fixed Q, has been studied extensively in the last ten years. This problem, related to those referred to in the Computational Geometry literature as the polygon placement problem and the motion planning problem, has been attacked using several assumptions about the kind of polygons and motions allowed. Here we briefly mention some of these results.

Suppose P and Q are p-gon and q-gon respectively; P can move while Q remains fixed. In 1983 Chazelle ([CH]) proposed an $O(pq^2)$ algorithm for the case where Q is convex and it is possible for P to translate and rotate, and gave an $O(p+q)$ procedure valid if P is restricted to translate. Also assuming just translations appeared in [BFM] an algorithm that runs in $O(pq\ log\ p)$ when both P and Q are rectilinearly convex polygons and, another restricted problem that considered P as convex was solved in [F] using $O(pq\ log\ pq)$. For both P and Q being non convex and even not connected, Avnaim and Boissonnat proposed in [AB] an $O(p^2q^2\ log\ pq)$ algorithm in the case of only translations, and showed that the method can be generalized to an $O(p^3q^3\ log\ pq)$ procedure when rotations are also possible. Other versions of the polygon containment problem (PCP) have been studied. See for example [CK] and [AIIT].

If the size of the output in each specific case is taken into account, it might be said that most of the above-mentioned results are nearly optimal. For example, when both P and Q are rectilinearly convex polygons, it was proved in [BFM] that the boundary of the set of feasible placements of P inside Q is a rectilinearly convex polygonal region

that could reach $\Omega(pq)$ edges, while the algorithm used there to obtain that region is $O(pq\ log\ p)$. Another close to optimal result can be found in [AB] as the solution may consist of $\Omega(p^3q^3)$ faces when translations as well as rotations are allowed. Except for the case in which Q is convex and P can only translate, we are not aware of either the existence of an optimal in the worst case algorithm for any of the above-stated problems or a proof of the non existence of such optimal algorithms.

Here a procedure is presented which shows that the problem of computing the path that a rectilinear non decreasing polygonal takes as it slides along the edges of another one can be reduced to the problem of sorting sets of numbers of the form $X+Y$. That result enables us to say that the PCP under translation in the case of rectilinearly convex polygons can be also reduced to sort $X+Y$. For this problem of sorting, the question of how much computation time is really needed is still open.

2 Preliminaries

In what follows, we denote the x-coordinate and the y-coordinate of a point p in the plane as $p.x$ and $p.y$ respectively.

Let A and B be rectilinear polygonals, i.e. with all their edges being horizontal or vertical and described through the list of their vertices given in order (every vertex and its succesor are the extremes of an edge). Suppose that A (B) refers to O_A (O_B), origin of the coordinate system. We will consider that B and its coordinate system are fixed and A with its coordinate system can translate.

Let us think of A and B as non decreasing polygonals, that is, if both v and u are vertices in one of them and v precedes u in the list of vertices then $v.x \leq u.x$ and $v.y < u.y$, or $v.x < u.x$ and $v.y \leq u.y$. We suppose that the bottommost edge of B (the topmost edge of A) is horizontal and its left (right) extreme is in some point in the infinite. On the other hand, the rightmost edge of B (the leftmost edge of A) is vertical and its upper (lower) extreme is also in some point in the infinite.

Let's denote the i^{th} vertex in one of the polygonals as v_i. We say that v_i is *convex* if the clockwise angle between segments $\overline{v_{i-1}v_i}$ and $\overline{v_iv_{i+1}}$ is 90°, and *concave* if that angle is 270°.

It is clear that B would be completely described through the list of its concave vertices as well as A through its convex vertices. Therefore, from now we assume that B is represented by the concave vertices b_1, b_2, ..., b_n and A by the convex vertices a_1, a_2, ..., a_m

We wish to obtain the vertices of the rectilinear polygonal that O_A would describe in the fixed coordinate system as A slides along the edges of B. In other words, we want a description of all the positions that O_A would reach if, beginning in some point in the

infinite with the topmost edge of A "touching" the bottommost edge of B, A translates to the right until going upward without intersecting B is possible, then A translates in that direction until the distance between some pair of horizontal segments is zero, etc. Notice that in essence, what we want is to solve a polygon containment problem in which both regions are not bounded.

For a certain position O of O_A referred to O_B, the coordinates of the vertex a_j ($1 \leq j \leq m$) in the fixed coordinate system is $(O.x+a_j.x, O.y+a_j.y)$ that we will denote as a_j^o.

3 The algorithm

Initially suppose O_A is placed in a position O so that there is contact between b_1 and a_m, that is, $a_m^o = b_1$.

Let's consider the set $D_V = \{(b_i, a_j^o)_V / 1 \leq i \leq n, 1 \leq j \leq m\}$ where the pair $(b_i, a_j^o)_V$ denotes the difference $b_i.y - a_j^o.y$ and the set $D_H = \{(b_k, a_l^o)_H / 1 \leq k \leq n, 1 \leq l \leq m\}$, where $(b_k, a_l^o)_H$ denotes $b_k.x - a_l^o.x$. For the sake of simplicity we assume that no two pairs in D_V are the same, i.e. there is not i, j, r, s, $1 \leq i, r \leq n$, $1 \leq j, s \leq m$, such that $(b_i, a_j^o)_V = (b_r, a_s^o)_V$. We assume a similar condition for D_H. These limitations result in no loss of generality.

Loosely speaking, our algorithm computes all contacts (in the order they take place) between vertices in B and vertices in A as the latter translates. Note that, if for a certain location of O_A, p, b_i contacts a_j, and we know that the next contact as A slides first upward and after to the right is b_r and a_s, then the next two positions of O_A are $(p.x, p.y + (b_r.y - a_s^p.y))$ and $(p.x + (b_r.x - a_s^p.x), p.y + (b_r.y - a_s^p.y))$, where the last one is the position for O_A in which b_r contacts a_s. This observation tell us how to obtain all the positions of O_A.

So, the problem is to obtain b_r and a_s. Let D_V^* and D_H^* be the sorted sets D_V and D_H respectively.

Lemma

If b_i touches a_j ($1 \leq i \leq n$, $1 \leq j \leq m$) for a certain location p of O_A then, the next contact as A slides upward and to the right is between b_r and a_s if, $(b_r, a_s^o)_V$ is the next pair in D_V^* following $(b_i, a_j^o)_V$ on the condition that its corresponding pair $(b_r, a_s^o)_H$ is located in D_H^* following $(b_i, a_j^o)_H$.

Proof

It is clear that the pair we are looking for must succeed $(b_i, a_j^o)_V$ in D_V^*. Let $(b_k, a_l^o)_H$ be a pair preceding $(b_i, a_j^o)_H$ in D_H^*. Then it is easy to realize that $b_k.x < a_l^p.x$ so it is impossible for a_l to make contact with b_k. □

The previous lemma enables us to design the following algorithm to compute the set of positions which O_A can be at. In the algorithm it is considered that there is a label associated to each element in D_H^* which can hold the values *unvisited* (there could be a contact) or *visited* (definitely no contact in the future). The contacts will be stored in L.

Procedure Compute-Contacts
begin
  Compute D_V and D_H;
  Compute D_V^* and D_H^*;
  Mark every pair in D_H^* as *unvisited*;
  Move A until a_m contact b_1 and mark as *visited* $(b_1, a_m^o)_H$ in D_H^*;
  $L \leftarrow (b_1, a_m)$;
  For every pair $(b_i, a_j^o)_V$ in D_V^* **do**
    if $(b_i, a_j^o)_H$ in D_H^* is *visited* **then** go to analysis next pair in D_V^*
    else
    begin
      $L \leftarrow (b_i, a_j)$;
      Mark $(b_i, a_j^o)_H$ as *visited* as well as every pair in D_H^* that precedes it down to the last marked pair
    end
end

We should notice that the running time of the previous procedure is dominated by sorting. If we use a standard sorting algorithm an $O(nm\ log\ m)$ upper bound would be obtained, which equals the one showed in [BFM]. Could D_V and D_H be sorted in less time, taking into consideration the particular structure of these sets?. The problem of sorting $X + Y$, where X and Y are sets of real numbers, has been studied before ([HPSS], [FR] and [LA]). It was proved in [LA] that, for two given sequences of numbers $(x_i)_{1\leq i\leq N}$ and $(y_j)_{1\leq j\leq N}$, there exists an algorithm to compute the N^2 sums $(x_i + y_j)_{1\leq i,j\leq N}$ in $O(N^2)$ comparisons. In fact, such an algorithm was presented there, but unfortunately, its performance was analized just in terms of comparisons and the existence of an algorithm with a similar bound in the case of a more general study remains still unknown.

The above procedure shows that the problem of sliding a rectilinear non decreasing polygonal of m edges along the n edges of another one can be reduced to sort $X + Y$, X and Y with n and m elements respectively; if there is an $O(nm)$ algorithm for the last problem then the solution for the former could be also obtained in $O(nm)$. On the other hand, the polygon containment problem under translation in the case of rectilinearly convex polygons can be reduced to compute such a sliding-path ([BFM]) hence

our algorithm proves the reducibility of that specific PCP to sort $X + Y$. More precisely, suppose Q, P are rectilinearly convex polygons with n, m vertices respectively. Determining all the positions at which P is contained by Q can be done within the same bound as sorting $X + Y$.

4 Conclusions

We have shown that the problem of sliding a non decreasing rectilinear polygonal along the edges of another one can be reduced to solve the problem of sorting $X + Y$, which in fact proves that the PCP under translation is also reduced to sort $X + Y$ when both polygons are rectilinearly convex. As determining lower bounds for both problems has yet to be solved, it will be interesting to see if the contrary relation holds true in further research.

5 References

[AB] F. Avnaim, J.D. Boissonnat, "Polygon placement under translation and rotation", *Proceedings of the STACS 1988*, Lectures Notes in Computer Science 294, Springer-Verlag, 1988, pp. 322-333.

[AIIT] H. Aonuma, H. Imai, K. Imai, T. Tokuyama, "Maximin location of convex objects in a polygon and related dynamic Voronoi diagrams", *Proceedings of the 6th ACM Symposium on Computational Geometry*, 1990, pp. 225-234.

[BFM] B.S. Baker,S.J. Fortune, S.R. Mahaney, "Polygon containment under translation", *Journal of Algorithms* **7** (1986), pp. 532-548.

[CH] B.Chazelle, "The polygon containment problem", *Advances of Computing Research*, Vol. 1, JAI Press Inc, Greenwich, Connecticut, 1983, pp. 1-33.

[CK] L.P. Chew, K. Kedem, "Placing the largest similar copy of a convex polygon among polygonal obstacles", *Proceedings of the 5th ACM Symposium on Computational Geometry*, 1989, pp. 167-173.

[F] S.J. Fortune, "A fast algorithm for polygon containment by translation", *Proceedings of the 12th ICALP*, Lecture Notes in Computer Science 194, Springer-Verlag, 1985, pp.189-198.

[FR] M.L. Fredman, "How good is the information theory about sorting?", *Theoretical Computer Science* 1 (1976), pp. 355-361.

[HPSS] L.H. Harper, T.H. Payne, J.E. Savage, E. Straus, "Sorting X+Y", *Communications of the ACM*, June 1975, Vol. 18, Number 6, pp. 347-349.

[LA] J.L. Lambert, "Sorting the sums $(x_i + y_j)$ in $O(n^2)$ comparisons", *Proceedings of the STACS 1990*, Lectures notes in Computer Science 415, Springer-Verlag, 1990, pp 195-206.