Univ. u Novom Sadu
Zb. Rad. Prirod.–Mat. Fak.
Ser. Mat. 23, 1 (1993), 179 - 188

Review of Research
Faculty of Science
Mathematics Series

AN ALGORITHM FOR CONSTRUCTING THE VORONOI DIAGRAMS IN PLANE

Miloš Racković, Dušan Surla[1]
Institute of Mathematics, University of Novi Sad
Trg Dositeja Obradovića 4, 21000 Novi Sad, Yugoslavia

Abstract

The presented algorithm for constructing the Voronoi diagrams in plane for a finite set of points S consists of two parts. In the first part the Voronoi polygons are constructed for the points of the convex hull of S, while in the second part, these polygons are constructed for the points of the interior region of the convex hull. The algorithm has been implemented in Pascal.

AMS Mathematics Subject Classification (1991): 68C05.
Key words and phrases: Voronoi diagrams, computational geometry.

1. Introduction

Let S be a set of points $q_1, q_2, ..., q_N$ in plane. The Voronoi polygon is a set of all points in the plane whose distance from the point q_i is shorter than from any other point belonging to S. This polygon is also known as the Dirichlet region, the Thiessen polygon, or the Winger-Seitz cell. The Voronoi diagram is a set of all the Voronoi polygons.

There are three main aspects of study of the Voronoi diagrams, and they are concerned with their mathematical properties [1, 2, 3, 4], their

[1]The work is supported by Science Fund of Serbia, grant no. 0413 through Institute of Mathematics, Novi Sad

construction [3, 5, 6, 7, 8], and the different generalizations and applications [9, 10]. An extensive survey of the literature covering all these aspects has been given recently [11].

The objective of the present work is to develop an effective algorithm for constructing the Voronoi diagrams. As first, a convex hull for the set of points S is formed. Then, two algorithms are formed: one for constructing the Voronoi diagrams for the points belonging to the convex hull, and the other for the points belonging to the interior region of the hull. The algorithmic structures have been implemented in Pascal, and a test example is presented.

2. The Voronoi diagrams and their properties

For the adopted metric $d(x,y)$ between the points x and y, the Voronoi polygon is a set of points in the plane: $\{x|\forall j : d(x,q_i) \leq d(x,q_j\}$. In the present paper we consider the construction of the Voronoi diagrams with the L_2-metric:

$$d(x,y) = ((x_1 - y_1)^2 + (x_2 - y_2)^2)^{1/2}$$

Let us denote by $B(q_i,q_j)$ the bisector between the points q_i and q_j.

The half-plane $H(q_i,q_j)$ is one of the two regions obtained on dividing the plane by the bisector $B(q_i,q_j)$ which contains the point q_i.

The Voronoi polygon $V(q_i)$ is the set of points in a plane whose distances from the point q_i are shorter than the distance from any other point $q_j, j \in \{1,2,...,i-1,i+1,...,N\}$, i.e.

$$V(q_i) = \bigcap_{j \neq i} H(q_i,q_j)$$

The Voronoi diagram $Vor(S)$ is a set of the Voronoi polygons $V(q_i), i \in \{1,2,...,N\}$.

The segments and terminal points of the Voronoi diagram are called the Voronoi edges and Voronoi vertices, respectively.

Under the assumption that there are no four points of the set S belonging to the one and the same circumference, the following can be proved [1]:

Each vertex of the Voronoi diagram is the intersection of exactly three edges of the diagram.

The polygon $V(q_i)$ is unbounded if and only if the point q_i belongs to the convex hull of S.

The polygons $V(q_i)$ with L_2-metric are convex polygons.

It has been shown [1] that the computational complexity of the constructing the Voronoi diagram is $O(NlogN)$, and it is optimal.

3. The algorithm

The construction of $V(q_i)$ for the points of the convex hull of S consists of the following. Let q_l and q_k be the neighboring points to the point q_i on the convex hull of S. Then, on the basis of the properties of the Voronoi diagrams it follows that the unbounded edges of $V(q_i)$ are lying on the bisectors $B(q_i, q_l)$ and $B(q_i, q_k)$. Let r be the intersection of these two bisectors. Then, the Voronoi polygon is represented by an open list of three points. The first point is located in the "infinity" on the unbounded edge, the second point is r, and the third point is in the "infinity" on the other unbounded edge. If the given bisectors are parallel, then there is a $B(q_i, q_j)$ which will intersect them (q_j belongs to the convex hull of S; if q_j does not exist, then all the points of S are collinear). In this case, the initial polygon is formed in a way analogous to that of the preceding case, but now of four points.

If the intersection of one of the subsequent and the initial polygon is not an empty set, this bisector divides the polygon in two polygons. The new polygon is one which is unbounded. The procedure for determining a new polygon is of the form:

```
procedure hull(var pr:pointer; c,d:point);
(* pr  - the pointer to the list representing  the
         Voronoi polygon that has to be updated by
         the bisector.
   c,d - the points lying on the bisector for
         updating the polygon. *)
begin
  p:=pr; first:=true;
  while (p^.next <> nil) do
    if (intersect(seg(p^.po,p^.next^.po),line(c,d))<>0) then
      begin
        Determination  of the  intersection  point
```

```
    if (intersect(seg(p^.po,p^.next^.po),line(c,d))<>0) then
      begin
        Determination of the intersection point
        of the segment and the straight line. If
        this point is different from the terminal
        points of the segment, then it is inserted
        into the corresponding place in the list;
        if first then
          begin
            first:=false; p:=p^.next
          end
        else
          begin
            Exclusion of the part between the two
            newly obtained points;
            exit
          end
      end
    else
      p:=p^.next
end; (* hull *)
```

The construction of $V(q_i)$ when q_i belongs to the interior region of the convex hull of S consists of the following. First, the three points q_l, q_m and q_n, belonging to the convex hull of S, are determined so that q_i belongs to the interior region of the triangle determined by the points q_l, q_m and q_n. The initial polygon is formed in the form of a circular list of three points. The first point is the intersection of the bisectors $B(q_i, q_l)$ and $B(q_i, q_m)$, the second of $B(q_i, q_m)$ and $B(q_i, q_n)$, and the third one is the intersection of $B(q_i, q_n)$ and $B(q_i, q_l)$.

If there are no points q_l, q_m and q_n, the convex hull of S is determined by four points. In this case the initial polygon for the point q_i is formed in a way analogous to that in the preceding case, but now it includes four points.

If the intersection of the subsequent bisectors and the initial polygon is not an empty set, the bisector divides the polygon in two polygons. The new polygon is one of the two polygons in which the point q_i is found. The procedure for determining the new polygon is of the form:

```
procedure intreg(var pr:pointer; a,c,d:point);
(* pr  - the pointer to the circular list representing
         the polygon.
   c,d - the points lying on the bisector.
   a   - the point for which the polygon is
         determined. *)
begin
  p:=pr; first:=true;
  repeat
    if (intersect(seg(p^.po,p^.next^.po),line(c,d))<>0) then
      begin
        Determination of the intersection point and
        its insertion  into the corresponding place
        in the circular list;
        if first then
          begin
            first:=false;
            p:=p^.next
          end
        else
          begin
            Exclusion of the part between the two newly
            obtained  points  from the circular list so
            that the point a remains inside the polygon
            formed  by  the  remaining  points from the
            circular list;
            exit
          end
      end
    else
      p:=p^.next
  until p=pr
end; (* intreg *)
```

The structure of the main program is of the form:

nph,npi - the numbers of points on the convex hull and in the interior region of the hull,respectively.

ph,pi - the respective arrays of points of the hull and in the interior

region of the hull.

pol.hu,pol.in - the respective arrays of pointers to the lists representing the Voronoi polygons of the points belonging to the hull and the points in the interior region of the hull.

```
begin
  for i:=1 to nph do
    begin
      ok:=false;
      Determination of the preceding and the subsequent
      point for ph[i], ph[pt] and ph[st];
      bisect(ph[i],ph[st],a,b); bisect(ph[i],ph[pt],c,d);
(* procedure bisect  determines two points on the bisector
   of the  segment which  is  determined by  the first two
   arguments of the procedure and their values presents in
   two other arguments of the procedure *)
      if det(ph[i],ph[pt],ph[st]) <> 0 then
(* det=0 if these three points are on the same straight
   line *)
        begin
          intpoint(a,b,c,d,intp);
(* in  intp we obtain  the  intersection point for the two
   straight lines represented by the points a,b and c,d *)
          Determination  of the infinity point ai on the
          straight line passing through a and b;
          Determination  of the infinity point ci on the
          straight line passing through c and d;
          Generation of the list with point ai, intp and
          ci in pol.hu[i];
          ok:=true
        end;
      for j:=1 to nph do
        if (j <> i ) and (j <> pt) and (j <> st) then
          begin
            if not ok then
              begin
                if det(ph[i],ph[j],ph[st]) <> 0 then
                  begin
                    bisect(ph[i],ph[j],e,f);
```

```
                    intpoint(a,b,e,f,intp);
                    intpoint(c,d,e,f,intp1);
                    Determination of the infinity point ai;
                    Determination of the infinity point bi;
                    Generation  of  the list with point ai,
                    intp, intp1 and ci in pol.hu[i];
                    ok:=true
                  end
              end
            else
              begin
                bisect(ph[i],ph[j],a,b); hull(pol.hu[i],a,b)
              end
          end;
        for j:=1 to npi do
          begin
            bisect(ph[i],ph[j],a,b); hull(pol.hu[i],a,b)
          end
    end;
  for i:=1 to npi do
    begin
      Determination of the points of the initial polygon;
      Generation  of  the  circular list of the points of
      the initial polygon in pol.in[i];
      if (the list has three points) then
        for j:=1 to nph do
          if (j <> prt) and (j <> drt) and (j <> trt) then
(* prt, drt and trt are the points defining the initial
   polygon *)
            begin
              bisect(ph[i],ph[j],a,b);
              intreg(pol.in[i],pi[i],a,b)
            end;
      for j:=1 to npi do
        if j <> i then
          begin
            bisect(ph[i],ph[j],a,b);
            intreg(pol.in[i],pi[i],a,b)
          end
```

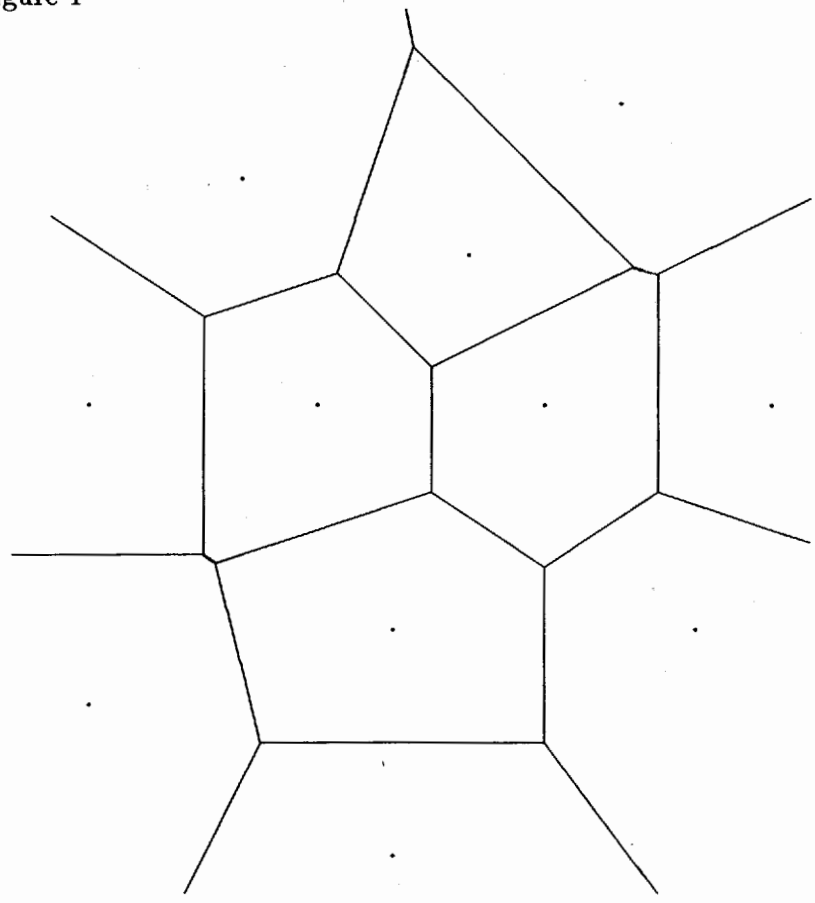
```
    end
end.
```

4. Numerical example

Figure 1 shows the Voronoi diagram for the following set of points:

$$S = \{ (2,4),(6,2),(10,5),(11,8),(9,12),(4,11),(2,8), (6,5),(8,8),(7,10),(5,8)\}$$

Figure 1



5. Conclusion

In computer construction of the Voronoi diagram the main problem is the determination of the unbounded edges of the diagram. Because of that the algorithm proposed deals first with the determination of the straight lines on which the unbounded edges are lying. On the basis of the properties of the Voronoi diagrams these straight lines coincide with the bisectors of the neighboring points belonging to the Voronoi diagram is constructed.

References

[1] Preparata, F., Shamos, M., Computational Geometry, An Introduction, Springer-Verlag, (1984).

[2] Chew L.P., Drysdale, R.L., Voronoi diagrams based on convex distance functions, Proc. 1-st Ann. ACM Symp. Computational Geometry (1985), 235-244.

[3] Lee, D.T., Two-dimensional Voronoi diagrams in the Lp-metric, J. ACM 27 (1980), 604-618.

[4] Gowda I. G., Kirkpatrick D. G., Lee D. T., Naamad A., Dynamic Voronoi diagrams, IEEE Trans. Information Theory IT-29 (1983), 724-731.

[5] Fortune, S., A sweepline algorithm for Voronoi diagrams, Algorithmica 2 (1987), 153-174.

[6] Ohya T., Iri M., Murota K., A fast Voronoi diagram algorithm with quaternary tree bucketing, Inf. Process. Lett. 18 (1984), 227-231.

[7] Dwyer R. A., A faster divide-and-conquer algorithm for constructing Delanay triangulations, Algorithmica 2 (1987), 137-151.

[8] Surla D., Ivanović M., An algorithm for constructing the Voronoi diagram, Proc. Computer at the university, Cavtat (1989), 11.10.1-11.10.6.

[9] O'Dunlaing C., Sharir M., Yap C. K., Generalized Voronoi diagrams for moving a ladder: II. efficient constructing of the diagram, Algorithmica 2 (1987), 27-59.

[10] Yap C. K., An $O(nlogn)$ algorithm for the Voronoi diagram of a set of simple curve segments, Discrete Comput. Geometry 2 (1987), 365-393.

[11] Aurenhamer F., Voronoi diagram - a survey, Rep. 236, IIG-TU Graz, Austria, (1988).

REZIME

ALGORITAM ZA KONSTRUKCIJU VORONOJEVIH DIAGRAMA U RAVNI

Algoritam za konstrukciju Voronojevih dijagrama za konačan skup S tačaka u ravni, sastoji se iz dva dela. Prvi se odnosi na konstrukciju Voronoivih poligona za tačke sa konveksnog omotača od S a drugi za tačke iz unutrašnje oblasti tog konveksnog omotača. Algoritam je implementiran u Pascalu.

Received by the editors June 13, 1990