An efficient algorithm for construction of the power diagram from the Voronoi diagram in the plane

Marina Gavrilova and Jon Rokne

Department of Computer Science, University of Calgary, Calgary, Alberta, T2N 1N4, Canada

Abstract. An algorithm for transforming a Voronoi diagram for point sites to a Voronoi diagram for weighted sites (power diagram) is presented. The algorithm was implemented and experiments suggest that the average running time of the algorithm and the space required are $O(N)$ in the plane. The worst case complexity can be shown to be bounded by $O(N^2)$ although it is conjectured that it is $O(N)$.

Keywords: computational complexity, computational geometry, Voronoi diagram, power diagram, Delaunay triangulation.

1. Introduction

This paper presents an algorithm for the efficient construction of the power diagram for set of circles S(C). The algorithm constructs the power diagram from the given Voronoi diagram for the set S(P) of point sites corresponding to the centers of the circles S(C). The algorithm is based on the sequential transformation of the Delaunay triangulation for the set of points S(P). The new set of edges is calculated in a specific order according to the weights and relative positions of the sites. The criteria of local changes in the Delaunay triangulation is also introduced. We assume that the circles S(C) do not intersect.

The algorithm is simple to understand and to implement, and it does not require complex computations. An experimental estimation of the total running time of the algorithm is also presented. We investigate distributions of the given point sites and the radii of the circles as well as the shapes of the region where objects have been distributed.

2. Definitions and methods

Voronoi diagrams for sets of points in Euclidean space have been studied extensively in their original as well as in more generalized settings. For a finite set $M \subseteq E^d$, the *closest-point Voronoi diagram (VD) of M* associates to each $p \in M$ a convex region $V(p)$ (*Voronoi region*) of all points closer to *p* than to any other point in *M*. More formally, $V(p) = \{x \in E^d \mid d(x,p) < d(x,q), \forall q \in M - \{p\}\}$, where *d* denotes the Euclidean distance function. The extensive description of algorithms, properties and applications of the VD can be found in works by Aurenhammer [2], Preparata and Shamos [12], Mulmuley [8], O'Rourke [9], Okabe, Boots and Sugihara [10].

A generalization of the VD can be obtained using the concept of weighting the given sites. Each $p \in M$ is assigned a real number $w(p)$, the *weight* of *p*, and the distance from *p* to a point $x \in E^d$ is measured as a function of $d(x,p)$ and $w(p)$. Two functions are commonly considered. The first function $d(x,p) - w(p)$ leads to diagrams whose representations can be interpreted as Voronoi diagram for circles or spheres with center *p* and radius $w(p)$. The edges of Voronoi diagram are in this case represented by hyperbolic curves. The second function is the *power function* of a point $x \in E^d$ which can be defined as $pow(x,p) = (x-p)^T(x-p) - w(p)$ [1]. For $w(p) > 0$ $\sqrt{pow(x,p)}$ represents the distance from the sphere with radius $\sqrt{w(p)}$ around *p* to the point *x* outside the sphere along the tangent line through the point *x* (see Figure 1).

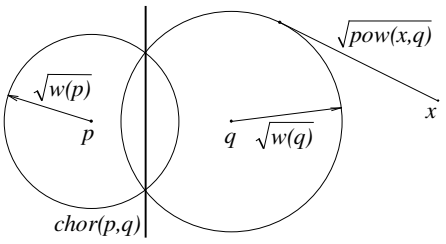

Figure 1. Power distance

Let a finite set of spheres S in E^d be given. For $s \in S$, the set

$$cell(s) = \{x \in E^d \mid pow(x,s) < pow(x,t), \forall t \in S - \{s\}\}$$

is called the *power cell* of *s* and the collection of all $cell(s)$ is the *power diagram* of *S*. In the plane, the locus of points of equal power with respect to two weighted sites *p* and *q* is called the *power line* of the corresponding circles. If the circles intersect, their power line passes through their points of intersection. In higher dimensions, the locus is a hyperplane in E^d, called the *chordale* of *p* and *q* or $chor(p,q)$. $chor(p,q)$ is orthogonal to the straight line connecting *p* and *q*. For $w(p) = w(q)$, $chor(p,q)$ degenerates to the bisecting hyperplane between *p* and *q*. Let $h(p,q)$ denote the halfspace bounded by $chor(p,q)$ and containing the point *p*. Then the power cell of *p* can be written as $cell(p) = \bigcap_{q \in S-\{p\}} h(p,q)$ [1].

There are a number of methods for constructing the power diagrams. Some algorithms for constructing power diagrams for circles in the plane is based on the divide-and-conquer technique ([6], [16]). Imai, Iri and Murota were among the first to develop a divide-and-conquer algorithm with the optimal running time $O(N \log N)$ in the plane and $O(N^2)$ in three dimensions [5]. The introduction of the sweep-plane algorithm for the constructing the first-order Voronoi diagrams for sets of points by Fortune provided a new approach for construction of the power diagrams for circles [3]. Rosenberger extended this approach to construct the *k*-order power diagram and attained optimal $O(k^2 N \log N)$ time-complexity and $O(kN)$ space-complexity [14]. Aurenhammer developed another approach, based on the duality of convex hulls and power diagrams of all orders in higher dimensions [1]. Hence, the complexity of algorithms for constructing convex hulls is carried to the power diagrams. Running times for the algorithms, based on this idea, are optimal $O(N \log N)$ for $d = 2$ [11], optimal $O\left(N^{\lceil d/2 \rceil}\right)$ for odd *d* [15] and for even *d*, $d \geq 3$ [17].

2. The algorithm

We are given a set of point sites in the plane with associated circles that do not intersect. We also assume that the Voronoi diagram for the non-weighted sites in the plane is known. We exclude degenerate cases, when four point sites are cocircular or four circles are cocircular in the sense of power metrics. We also assume that no 3 point sites lay on the same line.

The lower bound for the transformation of Voronoi diagram for point sites to the power diagram of weighted sites can be obtained from the estimated size of output as $O(N)$.

We will now describe an algorithm for the efficient construction of the power diagram for set of circles S(C). The algorithm demonstrate the possibility of construction such power diagram from the given Voronoi diagram for the set of the centers of the circles S(P) in the plane. It based on the sequential transformation of the Delaunay triangulation (DT) for set of points S(P). The new position of the edges are calculated in specified order according to the weights and relative positions of the sites.

The algorithm performs this transformation in $O(N)$ time experimentally. The space required is $O(N)$. It is conjectured that the algorithm has a running time strictly less than $O(N^2)$ since no example of a running time $O(N^2)$ has been found. A proof of this conjecture is not available, however. Later an example will be presented, where the algorithm in Euclidean metric requires $O(N^2)$ steps, but in power (Laguerre) metric still only $O(N)$ is needed.

The algorithm is based on the idea of "inflating" the point sites. Weights are first assigned to the sites. Then the sites are simultaneously inflated and the edges of the Voronoi diagram are translated to the new positions corresponding to the inflations. The edge corresponding to a pair of sites (a,b) is translated in parallel from the initial bisecting position to the new position (see Figure 2). The length of the edge is changed, and the edge can possibly disappear from the diagram.

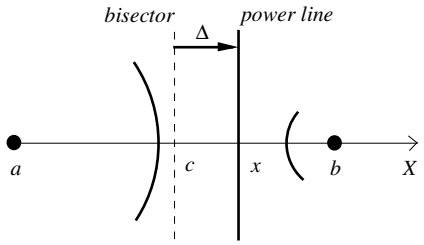

Figure 2. Adjustment of power line

Lemma 1

The bisector for point sites *a* and *b*, having being moved by the distance $\Delta = \frac{w_a - w_b}{2d}$ in parallel is the power bisector of weighted sites *a* and *b*, where w_a and w_b are the weights of the sites and where *d* is the distance between points *a* and *b*.

Proof

The proof is constructive in that we will construct the power bisector of *a* and *b* sites with weights w_a and w_b and show that it satisfies the suggested formula. Let us assume that the axis 0X is directed along the line segment *ab* and that the point *a* is the center of the coordinate system.

First we note that any point *x* on the power bisector satisfies the equation $pow(0,x) = pow(d,x)$ by definition. Therefore we have to solve the following equation:

$pow(x,a) = pow(x,b)$, where *x* represents the coordinate of the point on the intersection between segment *ab* and the power bisector. From this condition we obtain $x^2 - w_a = (d-x)^2 - w_b$. Resolving for *x* we get $x = \frac{d}{2} + \frac{w_a - w_b}{2d}$. If *c* is the coordinate of the bisector of the point sites, then the formula $x - c = \Delta = \frac{w_a - w_b}{2d}$ determines the distance between the bisector and the power bisector for the sites *a* and *b* (Figure 2)[1].

■

The transformation is done with each of the edges in the original Voronoi diagram. The new locations of the vertices are determined by the intersection of translated edges. In both the Voronoi and the power diagrams all of the vertices have degree three. Therefore, the new position for each vertex can be found by intersecting two lines containing any two of the translated edges (see Figure 3).

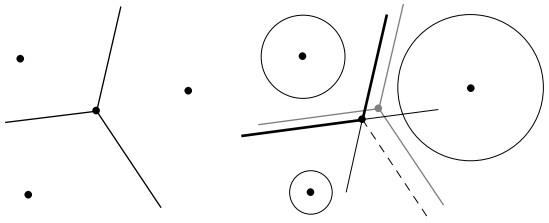
Figure 3. Transformation of Voronoi edges

The line containing the third edge must intersect the two other lines at the same point. This can be seen if we will consider the Voronoi diagram for these three sites only. Their power diagram must contain one vertex of degree three. Therefore, the third line must intersect the other two lines at the same point.

[1] The power line between two sites *a* and *b* always intersects the line segment *ab* under the assumption that the circles centered at *a* and *b* do not intersect.

Starting with a set of 4 points we assign a non-zero weight to the point *P* on the left in Figure 4(a), whereas in Figure 4(b) the point *P* in the lower part of the diagram is given a non-zero weight. For each edge the following two situations are therefore possible:

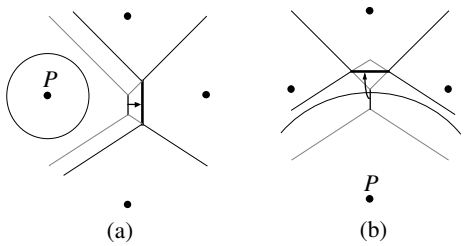

Figure 4. Modification of the Voronoi diagram

Situation *A*: the edge remains in the power diagram, i.e., the topological structure of Voronoi diagram remains the same (Figure 4a);

Situation *B*: the edge disappears from the power diagram and a new edge appears (Figure 4b).

In the situation *B* when the point *P* is inflated the length of the edge between the two points above *P* shrinks. Eventually it disappears and a new edge between *P* and point above appears. We might think of the original edge as shrinking towards zero, then growing "negative". This argument then shows that the unbounded edges on the perimeter of the Voronoi diagram can not disappear since they can not shrink from unbounded length to a negative length. The conclusion about which of two situations has occurred can be made based on the following lemma.

Lemma 2

Let $P_i = \{(x_i, y_i), w_i\}, i = 1..4$ be a set of points of the power diagram. Let define $D(P_1, P_2, P_3, P_4)$ as

$$D(P_1,P_2,P_3,P_4) = \frac{\det(\mathbf{A})}{\det(\mathbf{B})}, \text{ where } \mathbf{A} = \begin{bmatrix} x_1 & y_1 & x_1^2 + y_1^2 - w_1 & 1 \\ x_2 & y_2 & x_2^2 + y_2^2 - w_2 & 1 \\ x_3 & y_3 & x_3^2 + y_3^2 - w_3 & 1 \\ x_4 & y_4 & x_4^2 + y_4^2 - w_4 & 1 \end{bmatrix}, \mathbf{B} = \begin{bmatrix} x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \\ x_3 & y_3 & 1 \end{bmatrix}.$$

Then if

$D(P_1, P_2, P_3, P_4) < 0$ the edge will remain in power diagram (Situation *A*);

$D(P_1, P_2, P_3, P_4) > 0$ the edge will change its position (Situation *B*);

$D(P_1, P_2, P_3, P_4) = 0$ - degenerate case, when four circles are cocircular[2].

[2] In the implementation of the algorithm it is not necessary to perform division of determinants to calculate the sign of the expression $D(P_1, P_2, P_3, P_4)$. Only the signs of two determinants *A* and *B* have to be calculated in order to get the final answer. Determinant *B* is never equal to zero because of the assumption that no 3 points lay on the same line.

Proof

First of all, we will find the coordinates of the power vertex $X=(x,y)$ corresponding to the Delaunay triangle (P_1,P_2,P_3). The coordinates of this point satisfy the following system of equations: $pow(X,P_1)=pow(X,P_2)=pow(X,P_3)$. By solving this linear system, we find the coordinates of *X*:

$$x=-\frac{1}{2}\frac{\begin{vmatrix} y_1 & p_1 & 1 \\ y_2 & p_2 & 1 \\ y_3 & p_3 & 1 \end{vmatrix}}{\begin{vmatrix} x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \\ x_3 & y_3 & 1 \end{vmatrix}},\ y=\frac{1}{2}\frac{\begin{vmatrix} x_1 & p_1 & 1 \\ x_2 & p_2 & 1 \\ x_3 & p_3 & 1 \end{vmatrix}}{\begin{vmatrix} x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \\ x_3 & y_3 & 1 \end{vmatrix}}.$$

where $p_i=x_i^2+y_i^2-w_i, i=1..4$.

Then $D(P_1,P_2,P_3,P_4)$ can be defined as

$$D(P_1,P_2,P_3,P_4)=pow(X,P_1)-pow(X,P_4)=$$
$$(x-x_1)^2+(y-y_1)^2-w_1-\left((x-x_4)^2+(y-y_4)^2-w_4\right)=$$
$$\left(x_1^2+y_1^2-w_1\right)-\left(x_4^2+y_4^2-w_4\right)-2x(x_1-x_4)-2y(y_1-y_4)=$$
$$(p_1-p_4)+(x_1-x_4)\frac{\begin{vmatrix} y_1 & p_1 & 1 \\ y_2 & p_2 & 1 \\ y_3 & p_3 & 1 \end{vmatrix}}{\begin{vmatrix} x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \\ x_3 & y_3 & 1 \end{vmatrix}}-(y_1-y_4)\frac{\begin{vmatrix} x_1 & p_1 & 1 \\ x_2 & p_2 & 1 \\ x_3 & p_3 & 1 \end{vmatrix}}{\begin{vmatrix} x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \\ x_3 & y_3 & 1 \end{vmatrix}}=$$

$$\frac{(x_1-x_4)\begin{vmatrix} y_1 & p_1 & 1 \\ y_2 & p_2 & 1 \\ y_3 & p_3 & 1 \end{vmatrix}-(y_1-y_4)\begin{vmatrix} x_1 & p_1 & 1 \\ x_2 & p_2 & 1 \\ x_3 & p_3 & 1 \end{vmatrix}+(p_1-p_4)\begin{vmatrix} x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \\ x_3 & y_3 & 1 \end{vmatrix}}{\begin{vmatrix} x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \\ x_3 & y_3 & 1 \end{vmatrix}}=$$

$$=\frac{\begin{vmatrix} x_1-x_4 & y_1-y_4 & p_1-p_4 & 0 \\ x_1 & y_1 & p_1 & 1 \\ x_2 & y_2 & p_2 & 1 \\ x_3 & y_3 & p_3 & 1 \end{vmatrix}}{\begin{vmatrix} x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \\ x_3 & y_3 & 1 \end{vmatrix}}=\frac{\begin{vmatrix} x_1 & y_1 & p_1 & 1 \\ x_2 & y_2 & p_2 & 1 \\ x_3 & y_2 & p_3 & 1 \\ x_4 & y_4 & p_4 & 1 \end{vmatrix}}{\begin{vmatrix} x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \\ x_3 & y_3 & 1 \end{vmatrix}}.$$

In the situation when P_4 is closer to *X* than the other three points P_1, P_2 and P_3, i.e. $D(P_1,P_2,P_3,P_4)>0$, the edge disappears from the power diagram and a new edge appears (Figure 4b). Otherwise, the edge remains in the power diagram (Figure 4a). The degenerate case $D(P_1,P_2,P_3,P_4)=0$ may occur in which case the power diagram will contain a vertex of degree 4. This case will occur with probability close to 0.

■

3. Main scheme of the algorithm

The algorithm assumes that the Delaunay triangulation corresponding to the original Voronoi diagram has already been computed. This triangulation can be obtained from the Voronoi diagram in $O(N)$ steps. Then all the sites are simultaneously inflated and the local improvements (swaps) are sequentially performed on the Delaunay triangulation . For each internal edge in the triangulation $D(P_1, P_2, P_3, P_4)$ are calculated. If the value of $D(P_1, P_2, P_3, P_4)$ is positive then situation *B* occurs and the SWAP operation [13] will be performed on the triangulation (see Figure 5). If the value of $D(P_1, P_2, P_3, P_4)$ is negative (Situation *A*) then Delaunay triangulation should not be changed.

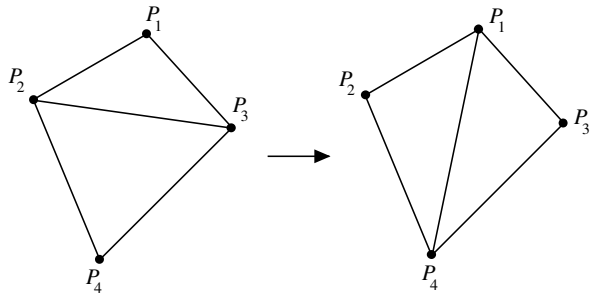

Figure 5. SWAP operation

Let us assume that the number of vertices in the original DT is N, then the number of edges is $m=O(N)$.

Algorithm description

Step 1. Put all edges of the Delaunay triangulation into the queue Q

Step 2. While Q is not empty do begin

extract a first edge *E* from the queue Q;

calculate the sign of $D(P_1, P_2, P_3, P_4) = 0$ according to *Lemma 2*;

if $D(P_1, P_2, P_3, P_4)$ then begin

perform SWAP operation on the edge *E*;

find four neighboring edges *E*1, *E*2, *E*3, *E*4 from the quadruple P_1, P_2, P_3, P_4;

remove *E*1, *E*2, *E*3, *E*4 from the Q (only those of them which are in the queue currently);

insert *E*1, *E*2, *E*3, *E*4 into the end of Q;

end;

end { while };

Step 3. Stop

■

The resulting triangulation is the power triangulation for given sites with additive weights. The calculation of the determinant and SWAP operation can be done in $O(1)$ time and the number of internal edges in the Delaunay triangulation is $O(N)$ in each step. The power diagram can be obtained from the constructed Delaunay triangulation in $O(N)$ time. The resulting triangulation is the power triangulation for the given sites with additive weights.

The complexity of the algorithm can be estimated as follows. The basic SWAP step can be done in $O(1)$ time since it involves the calculation of a constant size determinant. There are $O(N)$ internal edges in the Delaunay triangulation. Each particular SWAP on an internal edge can cause a propagating wave of at most $O(N)$ SWAPs. The worst case complexity is therefore bounded by $O(N^2)$. The experimental results of the next section show, however, that the number of SWAPs required in an actual computation is only a small

fraction of the total number of edges in the Delaunay triangulation. Since each edge has to be checked for a possible SWAP, the total number of steps is at least $O(N)$. However, the experiments suggest a conjecture that the total number of SWAPs required to perform the transformation is strictly less than $O(N^2)$ and possibly equal to $O(N)$. It has not been possible to either provide a proof of this conjecture or a counterexample showing $O(N^2)$ complexity.

The space complexity of the algorithm is $O(N)$, since the edge queue will not contain more than $O(N)$ edges at any moment of time. The Quadedge structure used in the algorithm also requires $O(N)$ space.

4. Implementation of the algorithm

The algorithm is implemented using the Quadedge data structure [4]. The initial Delaunay triangulation is constructed by the incremental algorithm described in [7]. The algorithm is implemented in Object Pascal in the Borland Delphi environment on PC. The program has a user-friendly interface and runs under Microsoft® Windows® 95. The experiments were conducted on a 486DX2/66 PC with 16 megabytes of RAM. The user of the program can create a new distribution of circles, modify an existing one or automatically generate a random distribution. There are three ways of specifying the distributions:

1) manual construction of the distribution in the integrated editor, by drawing circles with the mouse;
2) importing the distribution from the text file, where the coordinates of the centers and radii of circles are specified;
3) automated random generation of the distribution, where the user can specify the parameters of the random distribution, such as the number of circles, the distribution of their radii, and the shape of the distribution (square, circle, cross, annulus, or corners of the square).

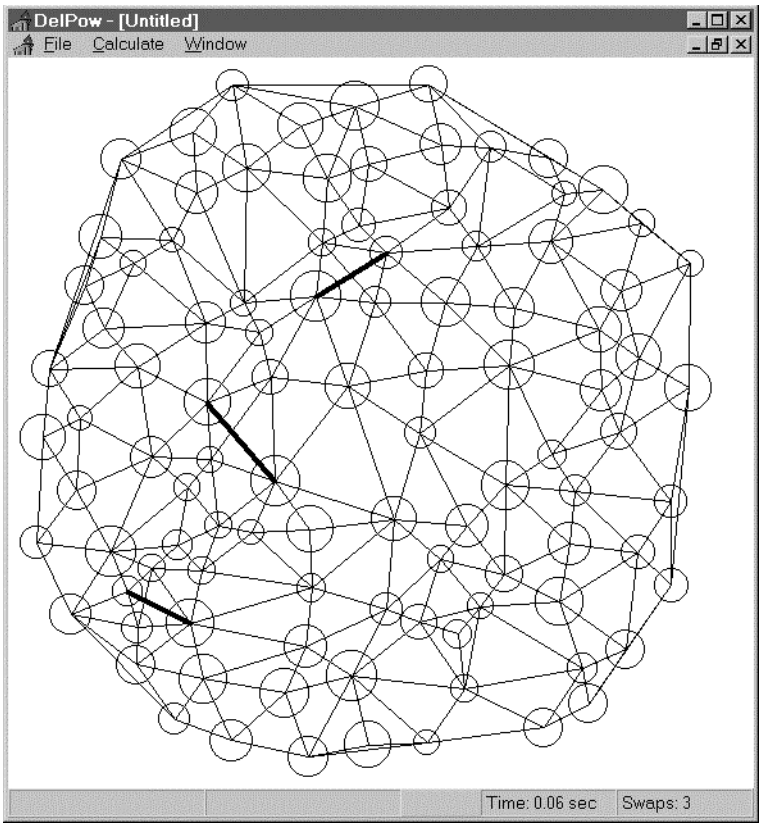

Figure 6. Program interface

After the desired model has been created, the Delaunay triangulation for point sites and the Delaunay triangulation in power metrics for the circles are constructed (see Figure 6). The number of SWAPs that

occurred during the transformation and the elapsed time are reported. The edges of the DT, where the SWAPs appeared, are highlighted by a different color. The distribution can be saved for the following tests or modifications.

5. Experimental results

In all the experiments the algorithm performed the transformation in $O(N)$ time. For most of the cases the experimental results show that only αN SWAP operations are required, where α is a small constant (see Figure 7). We experimented with random distributions of point sites over different shapes (square, circle, cross, annulus, and square corners), for different number of sites and for variable densities of circle distributions. No cases were encountered where the algorithm required more than $O(N)$ steps. While the experiments indicate that the bound for the running time is strictly less than $O(N^2)$, it appears that a proof of this conjecture is difficult to find.

In the first series of experiments the number of circles was gradually increased from 100 to 3000 per square, with a number of independent experiments conducted on each step. The density of the distribution was maintained as a constant and the number of the SWAP operations and required time for the transformation were calculated. The graph in the Figure 7 demonstrates that the number of SWAPs grows as a linear function of the number of point sites, gradually increasing from low average of 4 SWAPs for 100 circles to 50 SWAPs for a 1000 and to 150 for a 3000 circles.

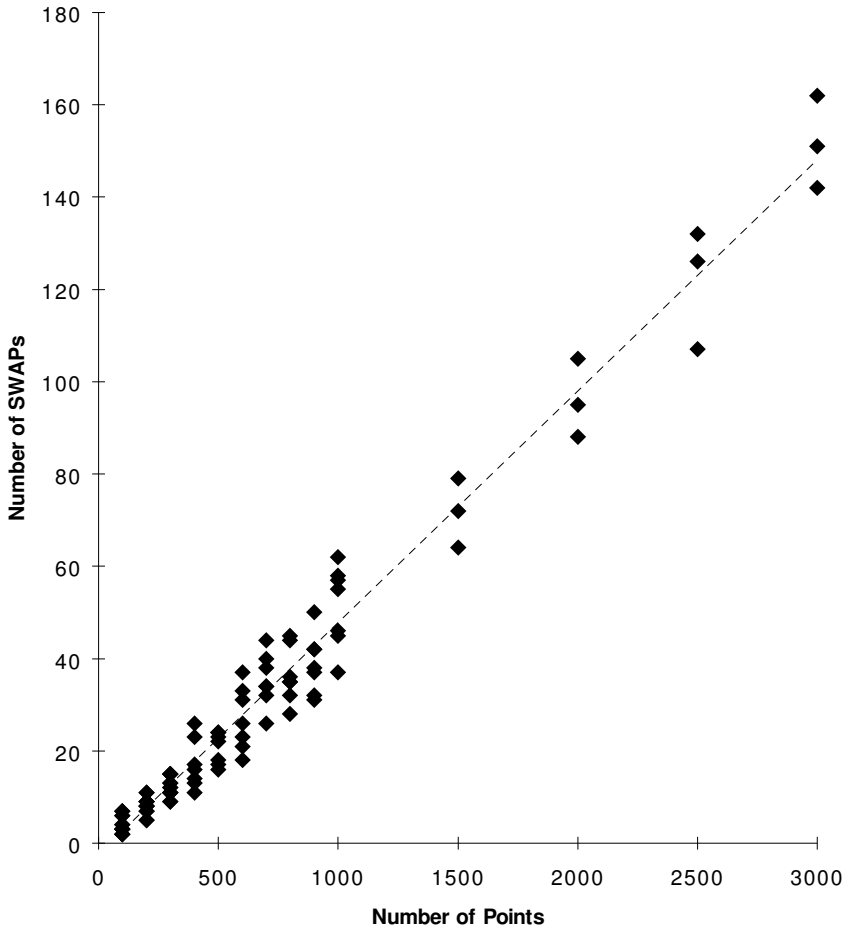

Figure 7. Number of SWAPs required

The time, required for the transformation, also grows as a linear function of the number of point sites. The following graph (Figure 8) shows the average time, based upon the number of the independent experiments for the same number of point sites versus the number of point sites. It takes about one tenth of a second to perform the transformation for a hundred circles and about a second for 300 circles on the PC described above. Then the time gradually grows to 8 sec per 1000 circles, increasing by about 1 sec per additional hundred objects. When a number of elements exceeds one thousand, the process slows down a bit and the time increases to 19 seconds for 2000 circles, and 35 seconds for 3000 circles. The first experiment demonstrated that the algorithm performed the transformations in $O(N)$ time with only αN SWAP operations required in total, where α is a small constant

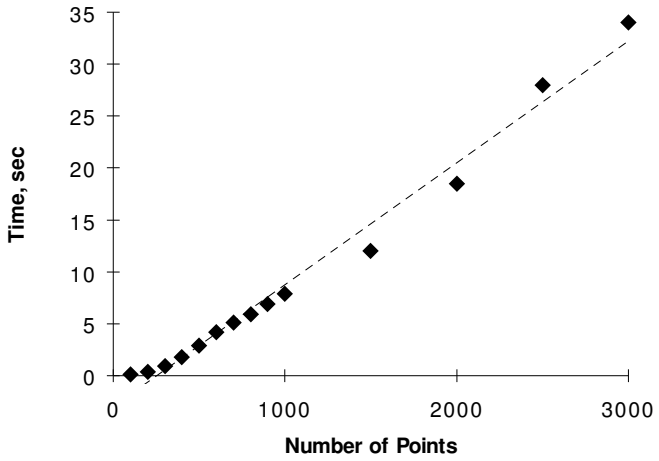

Figure 8. Elapsed time

In the second series of experiments we increased the density of the distribution by varying the number of circles with radii, ranging from 2.5 to 5, randomly distributed inside the square (see Figure 9). The number of SWAPs shows the growth in a slowly growing quadratic function depending of the increase of the number of circles from 100 to a 1000 (density increase). The number of SWAPs is still a small constant. As an example only 45 SWAPs were required for 1000 objects.

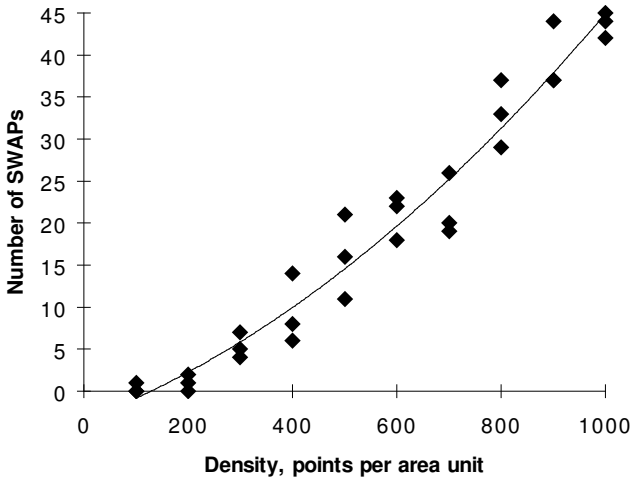

Figure 9. Density

In the third series of experiments we investigated the number of SWAP operations for different shapes: square, circle, cross, annulus, and square corners in each case for different number of points and densities. The results show that these numbers are very similar to the previous ones. We, therefore, conclude experimentally that the shape of the distribution does not influence the number of SWAPs.

6. Extensions of the algorithm

One can note, that the algorithm can be tailored to convert Delaunay triangulation for points into a Delaunay triangulation for circles in other metrics, for example, the Euclidean metrics. The only part that has to be changed is the formula for the INCIRCLE test. The average case performance of the algorithm remains the same, i. e. $O(N)$. However, there exists an example, where $O(N^2)$ SWAPs are required to perform the transformation (see Figure 10). For the same example in the power metric the Delaunay triangulation does not change at all (no SWAPs are required).

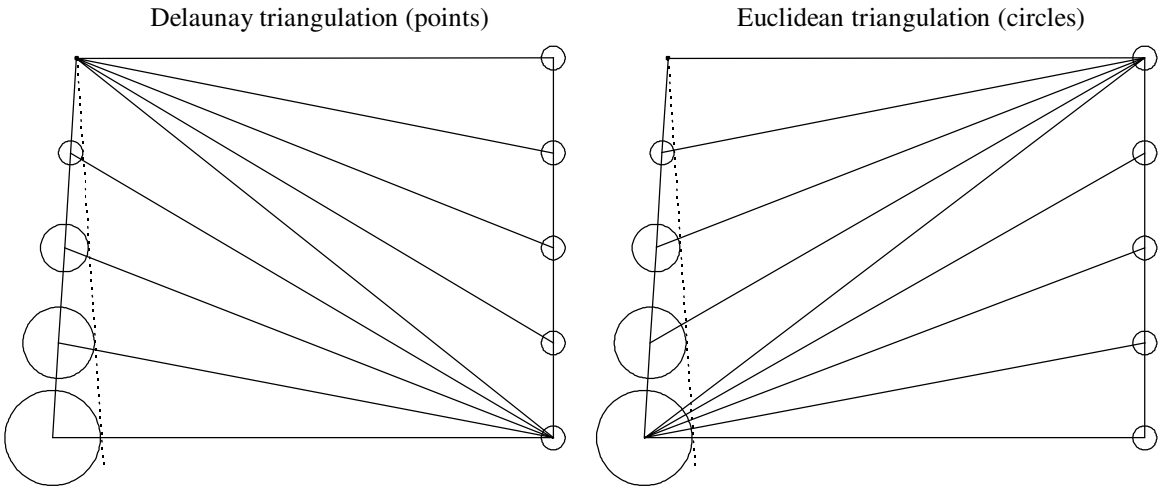

Figure 10. $O(N^2)$ SWAPs are required for transformation for the Euclidean metrics

7. Conclusions

A new approach for construction of the power diagram in the plane is presented. It is based upon transforming the Delaunay triangulation for point sites in the plane into the power triangulation for weighted sites (circles). Experiments show that the algorithm requires $O(N)$ time and $O(N)$ space in all cases tested.

8. References

[1] Aurenhammer, F. 1987. "Power Diagrams: Properties, Algorithms, and Applications," *SIAM Journal of Computing*, Vol. 16, No. 1: 78-96.

[2] Aurenhammer, F. 1991. "Voronoi Diagrams - A Survey of a Fundamental Geometric Data Structure," *ACM Computing Surveys*, Vol. 23, No. 3: 345-405.

[3] Fortune, S. 1987. "A Sweepline Algorithm for Voronoi Diagrams," *Algorithmica*, Vol. 2: 153-174.

[4] Guibas, L. and Stolfi, J. 1985. “Primitives for the manipulation of general subdivisions and the computation of Voronoi diagrams,” ACM Transactions on Graphics, 4(2): 74-123.

[5] Imai, H., Iri, M. and Murota, K. 1985. "Voronoi Diagram in the Laguerre Geometry and Its Applications," *SIAM Journal of Computing*, Vol. 14, No. 1: 93-105.

[6] Lee, D. and Drysdale, R. 1981. "Generalization of Voronoi Diagrams in the Plane," *SIAM Journal of Computing*, Vol. 10, No. 1: 73-87.

[7] Lischinski, D. 1994. “Incremental Delaunay Triangulation,” Graphics Gems, Academic Press.

[8] Mulmuley, K. 1994. “Computational geometry: An Introduction Through Randomized Algorithms,” Prentice-Hall, Inc. A Simon & Schuster Co., Englewood Cliffs, New Jersey, USA. 106-111.

[9] O'Rourke, J. 1994, "Computational Geometry in C," Cambridge University Press, Cambridge, New York, USA, 182-193.

[10] Okabe, A., Boots, B. and Sugihara, K. 1992, "Spatial Tessellations: Concepts and Applications of Voronoi Diagrams," John Wiley & Sons, Chichester, West Sussex, England, 205-208.

[11] Preparata, F. P. and Hong, S. 1977. "Convex Hulls of Finite Sets of Points in Two and Three Dimensions," *Communications of ACM*, Vol. 20: 87-93.

[12] Preparata, F. P. and Shamos, M. I. 1985. "Computational geometry An Introduction", Springer-Verlag New York Inc., New York, USA, 209-211.

[13] Roos, T. 1990. "Voronoi Diagrams over Dynamic Scenes (Extended Abstract)," In *Proceedings of 2nd Canadian Conference on Computational Geometry*: 209-213.

[14] Rosenberger, H. 1991. "Order-*k* Voronoi Diagrams of Sites with Additive Weights in the Plane," *Algorithmica*, Vol. 6: 490-521.

[15] Seidel, R. 1981. "A Convex Hull Algorithm Optimal for Point Sets in Even Dimensions," Report 81-14, University of British Columbia, Vancouver, Canada.

[16] Sharir, M. 1985. "Intersection and Closet-Pair Problems for a Set of Planar Disks," *SIAM Journal of Computing*, Vol. 14, No. 2: 448-468.

[17] Stewart, W.M.1990. "An Algorithm to Find the Facets of the Convex Hull in Higher Dimensions," In *Proceedings of the 2nd Canadian Conference in Computational Geometry*: 252-256.

[18] Voronoi, G. 1908. "Nouvelles applications des paramètres continues à la théorie des formes qudratiques, Deuxième Mémoire: Recherches sur les paralléloedres primitifs," *J. Reine Angew. Math*., Vol. 134: 198-287, Vol. 136: 67-181.