Available online at www.sciencedirect.com

SCIENCE DIRECT®

Information Processing Letters 97 (2006) 138–145

Information Processing Letters

www.elsevier.com/locate/ipl

Robustness of k-gon Voronoi diagram construction

Zhenming Chen [a,1], Evanthia Papadopoulou [b], Jinhui Xu [a,*,1]

[a] *Department of Computer Science and Engineering, State University of New York at Buffalo, Bell Hall 201, Buffalo, NY 14260, USA*
[b] *IBM T.J. Watson Research Center, P.O. Box 218, Yorktown Heights, NY 10598, USA*

Received 17 May 2005; received in revised form 4 October 2005; accepted 26 October 2005

Available online 18 November 2005

Communicated by F.Y.L. Chin

Abstract

In this paper, we present a plane sweep algorithm for constructing the Voronoi diagram of a set of non-crossing line segments in 2D space using a distance metric induced by a regular k-gon and study the robustness of the algorithm. Following the *algorithmic degree* model [G. Liotta, F.P. Preparata, R. Tamassia, Robust proximity queries: an illustration of degree-driven algorithm design, SIAM J. Comput. 28 (3) (1998) 864–889], we show that the Voronoi diagram of a set of arbitrarily oriented segments can be constructed with degree 14 for certain k-gon metrics (e.g., $k = 6, 8, 12$). For rectilinear segments or segments with slope $+1$ or -1, the degree reduces to 2. The algorithm is easy to implement and finds applications in VLSI layout.
© 2005 Elsevier B.V. All rights reserved.

Keywords: Voronoi diagram; Algorithmic degree; Computational geometry

1. Introduction

The Voronoi diagram of polygonal objects is a fundamental geometrical structure with numerous applications in diverse areas. Excellent surveys on the background, construction and applications of Voronoi diagram can be found in [1–3,15]. However, robustness issues associated with the construction of the diagram have made its use hard in practice [9,18]. In this paper we investigate the algorithmic *degree* of constructing the Voronoi diagram of segments under a regular k-gon distance metric as it can provide a more robust alternative to ordinary Euclidean Voronoi diagram for certain types of segments and values of k (e.g., $k = 8$).

The *degree* of an algorithm was introduced by Liotta et al. [11] to model its robustness. It reflects the lowest precision (i.e., the minimum number of bits) to which arithmetic calculations have to be executed in order to guarantee the topological correctness of the flow of the algorithm (i.e., if the input points are b-bits integers and the degree of the algorithm is d, then the least bit precision required is $db + \mathrm{O}(1)$). Our interest in the robustness of constructing the Voronoi diagram of segments under the k-gon distance metric is motivated by applications in VLSI layout as presented in [13]. Note that VLSI shapes are always in fixed orientations, and often consist only of axis parallel segments and segments of slope ± 1. Due to the enormous size of VLSI

[*] Corresponding author.
E-mail addresses: zchen4@cse.buffalo.edu (Z. Chen), evanthia@watson.ibm.com (E. Papadopoulou), jinhui@cse.buffalo.edu (J. Xu).
[1] The research of the first and third authors was supported in part by a faculty start-up fund from the CSE department, SUNY at Buffalo, and by an IBM faculty partnership award.

0020-0190/$ – see front matter © 2005 Elsevier B.V. All rights reserved.
doi:10.1016/j.ipl.2005.10.009

data, reasonable approximation, robustness and speed of computational methods are criteria far more important than 100% accuracy of the shape of Voronoi diagram.

The most basic operation in the construction of any Voronoi diagram is the so-called *incircle test*, which is the exact arithmetic computation to check if one point is inside the circumcircle of three given points. In the case of line segments and the Euclidean metric, Burnikel [4, 5] showed that the incircle test can be computed with degree 40. But this is rather high for practical applications. Note that the higher the degree of a *predicate* (i.e., a test used at a branch of the algorithm to determine the flow of control, such as incircle test), the greater the slow down due to the need for the use of arithmetic filters. In [13], the use of the L_∞ (resp. L_1) metric was proposed as a practical solution to the high degree problem associated with the construction of the Voronoi diagram of segments. It was shown that under the L_∞ metric, the incircle test for arbitrary line segments can be answered with degree 5 and the plane sweep paradigm [8] results in an algorithm of total degree 7. The degree can be reduced to 1 when the segments are in some fixed orientations.

Between the L_1 and the L_2 metrics, a family of metrics has been defined based on a regular k-gon, $k = 4, \ldots, \infty$ [17,16]. The L_1 metric is derived for $k = 4$ and the L_2 metric is reached as k approaches ∞. A similar family of metrics between the L_∞ and the L_2 metric can also be defined. It is thus natural to ask whether the k-gon metric can provide a sharper than the L_∞/L_1, yet robust, approximation to the L_2 Voronoi diagram. The *octagon* metric derived by an *isothetic* octagon (i.e., an octagon with axis parallel diagonals or axis parallel edges), captures very well the proximity information in VLSI designs where shapes consist typically of edges in axis parallel, $\pi/4, 3\pi/4$ orientations. In this paper, the degree of k-gon Voronoi diagram construction for certain k (e.g., 6, 8, 12) is studied.

The Voronoi diagram of segments, or more generally the Voronoi diagram of disjoint convex polygons, under general convex polygonal distances was studied in [12]. The case of points have been investigated earlier in [17, 6]. In [12] the combinatorial properties of this diagram were determined and a plane sweep method based on Fortune's paradigm [8] was given to construct a compact representation of the Voronoi diagram. With the compact representation, the actual Voronoi diagram can be retrieved in time proportional to its size.

In this paper we give a plane sweep algorithm to construct the actual Voronoi diagram with time complexity $O(nk \log nk)$, where n is the total number of segments and k is the size of the k-gon. Our algorithm has a slightly worse asymptotic time complexity than the one in [12] (i.e., $O(nk + n \log n \log k)$), but is simpler to implement. More interestingly, we show that the plane sweep algorithm has an algorithmic degree of at most 14 for a set of arbitrarily oriented segments under certain k-gon metrics (e.g., $k = 6, 8, 12$). For rectilinear segments or segments with slope ± 1, the degree reduces to 2.

We notice that all the above diagrams fall under the umbrella of abstract Voronoi diagrams [10] and techniques such as divide and conquer or randomized incremental construction can be applied once the computation of a bisector is available. Note that plane sweep methods are not available for abstract Voronoi diagrams. Plane sweep methods are typically vulnerable to robustness problems. However, as long as the associated robustness issues can be resolved, plane sweep is often preferable because of its simplicity.

2. k-gon Voronoi diagram

In this section, we discuss the concept of k-gon metric, the bisectors between points and segments, the plane sweep method used to construct the k-gon Voronoi diagram, and the sweeping.

In Minkowski geometry, any convex set P can be used to define a *convex distance function* between two points p and q as the amount that P needs to be scaled to include the vector $q - p$. The special case when P is a regular k-gon is termed as k-gon metric. The k-gon distance between two points p, q is the *size* of the smallest k-gon touching p and q, which can be naturally defined either in terms of the *diameter* or the *width* of P. The *diameter* is twice the radius which is the distance from the center of P to any of its vertices; The *width* of P is the distance between two parallel edges of P (assuming k is even). In this paper, we use diameter to derive a family of metrics whose unit circles are inscribed in the ordinary L_2 unit circle. Alternatively we could use the width to derive a class of metrics whose unit circles are circumscribed to the ordinary L_2 unit circle.

Since the orientation of k-gon could affect the distance function, we assume, unless stated otherwise, that the following orientation is used. For a k-gon P with radius r and centered at the origin o: $(0, 0)$, P is always oriented in such a way that one of its vertices coincides with the point $(r, 0)$. If we count the vertices counterclockwise, then the ith vertex v_i of P is $(r \cos i\theta, r \sin i\theta)$, where $\theta = 2\pi/k$. The edge e_i: (v_i, v_{i+1}) is supported by the line

$$l_i\colon\ \frac{y - r\sin i\theta}{x - r\cos i\theta} = \frac{\sin(i+1)\theta - \sin i\theta}{\cos(i+1)\theta - \cos i\theta}.$$

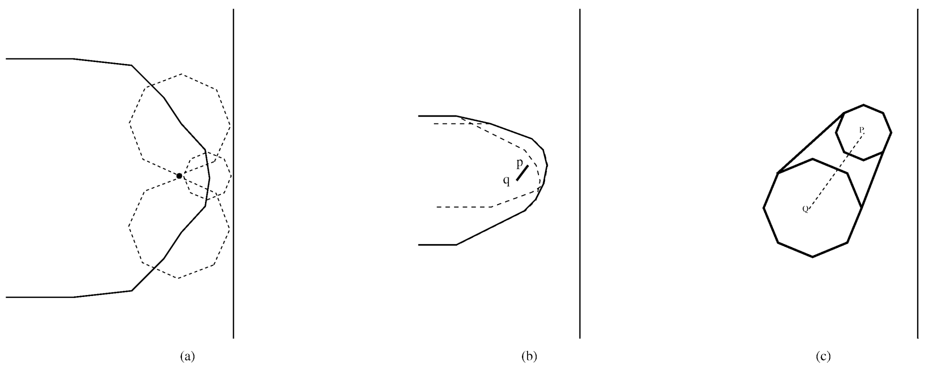

Fig. 1. (a) Bisector between a point and a vertical line (i.e., the sweepline). (b) Bisector between a segment and the sweepline. (c) Equidistant octagons and the convex hull.

Notice that for 3, 4, 6, 8, 12-gons, all the $\sin i\theta$, $\cos i\theta$ are multiple of either $\sqrt{2}$ or $\sqrt{3}$, and we will show that such property helps to reduce the degree of our algorithm.

Based on the aforementioned k-gon metric, we can define the k-gon bisector of a pair of geometric objects, such as a pair of points, a point and a segment, two segments, a point and a line, and a segment and a line. Fig. 1 gives a few examples of bisectors between two objects under the octagon metric. Fig. 1(b) shows the bisector (solid line) between a segment and the sweepline, which is derived from the bisectors (dotted line) between the two endpoints and the sweepline. Fig. 1(c) shows the octagons, equidistant to the two endpoints of a segment, and their convex hull. The distance from the octagon to the corresponding endpoint is half of that between the endpoint and the sweepline. Comparing to their counterparts under L_2 metric, k-gon bisectors have some nice properties: (1) Each k-gon bisector consists of only $O(k)$ line segments and no parabolic arc; (2) k-gon bisectors between a segment and the vertical sweepline are y-monotone.

Using these bisectors, especially the ones between a point or a segment and a vertical line, we can define the k-gon Voronoi diagram for a set of segments. Let G be a set of interior-disjoint 2D segments. The Voronoi diagram of G, denoted by $V(G)$, is a partition of the plane into regions, called *Voronoi cells*, such that each cell is associated with a segment of G. A point q belongs to the cell corresponding to $s_i \in G$ if and only if the k-gon distance from q to s_i is smaller than the distance from q to any other object in G, i.e., $d(q, s_i) < d(q, s_j)$ for each $s_j \in G$, $j \neq i$. In $V(G)$, a *Voronoi edge* e is a part of the bisector of the two objects in the neighboring cells of e, and a *Voronoi vertex* v is a point equidistant to three or more objects in G.

To construct the k-gon Voronoi diagram $V(G)$, we follow the commonly used plane sweep method [7,8,14]. Due to the special properties of the k-gon metric, the algorithm has some interesting features and yields a robust solution.

To facilitate the construction of the k-gon Voronoi diagram, we assume that there exists a big enough axis-aligned bounding box $\mathcal{B}$ which holds all the input segments and contains the topological structures (e.g., the Voronoi vertices and bounded edges) of $V(G)$. The sweepline L sweeps across the bounding box $\mathcal{B}$ from left to right, and the Voronoi diagram $V(G)$ is incrementally constructed during this sweeping process. The *beachline* is the lower envelop of the bisectors between the sweepline and all the swept segments (or subsegments).

The base case of the beachline is the bisector between a point p and the vertical sweepline L. For any point p at (x_p, y_p) and L at $x = x_l$, suppose $x_l > x_p$ and their distance is $d = x_l - x_p$. The bisector between p and L consists of $O(k)$ segments of the following form with each corresponding to an edge of the regular k-gon:

$$\frac{y - y_{v_i}}{x - x_{v_i}} = \frac{y_{v_{i+1}} - y_{v_i}}{x_{v_{i+1}} - x_{v_i}},$$

where

$$v_i = \left(x_p + \frac{d\cos i\theta}{1+\cos i\theta}, y_p + \frac{d\sin i\theta}{1+\cos i\theta}\right),$$
$$\theta = 2\pi/k.$$

The above equation can be rewritten as

$$y = A_i(x - x_p - dB_i) + y_p + dC_i, \quad i \neq k/2,$$

where

$$A_i = \frac{\sin(i+1)\theta(1+\cos i\theta) - \sin i\theta(1+\cos(i+1)\theta)}{\cos(i+1)\theta(1+\cos i\theta) - \cos i\theta(1+\cos(i+1)\theta)}$$

$$B_i = \frac{\cos i\theta}{1+\cos i\theta}, \quad \text{and} \quad C_i = \frac{\sin i\theta}{1+\cos i\theta}.$$

It is easy to see when $i = k/2$, the two components of the bisectors are actually horizontal rays of the following form, $y = y_p + (d\sin\theta)/(1-\cos\theta)$ and $y = y_p - (d\sin\theta)/(1-\cos\theta)$.

The bisector of a segment $s = (p, q)$ and the sweepline L can be derived from the two bisectors between the two endpoints of s and L, respectively. Let P_1 and P_2 be two regular k-gons that centered at the two endpoints p and q of s, respectively. Assume that the k vertices of P_1 and P_2 are all labeled counterclockwise, starting from their rightmost ones. Then there exist two numbers j and $j + k/2$ such that connecting the two jth vertices and the two $(j + k/2)$th vertices of P_1 and P_2, respectively, results in the convex hull of the two regular polygons. Since each regular polygon (Fig. 1(c)) is equidistant to its corresponding endpoint and the sweepline, it is easy to see that the index j (called ch-index) can be determined by the slope of the input segment. The line segment connecting the two jth vertices in the bisectors of p and q is also part of the bisector of s and the sweepline L: $x = t$. The equation of this segment is

$$\begin{aligned} &D_j(x - x_q) - d_x(y - y_q) \\ &\quad + d_q(d_x \sin j\theta - d_y \cos j\theta) = 0, \end{aligned}$$

where

$$\begin{aligned} D_j &= d_y(1 + \cos j\theta) - d_x \sin j\theta, \\ d_q &= t - x_q, \\ d_x &= x_p - x_q, \\ d_y &= y_p - y_q. \end{aligned}$$

Let G_t be the set of swept segments or subsegments by L at time t (i.e., L is at $x = t$). To compute the Voronoi diagram $V(G_t)$ of G_t, we dynamically maintain the set $B(G_t)$ of bisectors of L and the segments in G_t. The lower envelop (looking horizontally from $x = \infty$) of $B(G_t)$ (called *beachline*) contains two types vertices (called *breakpoints*): (1) vertices of the bisectors of the segments in G_t and (2) intersections of different bisectors. To compute the Voronoi diagram, it is sufficient to keep track of the change of the beachline, determine the vertices and edges of $V(G_t)$, and finalize them as part of $V(G)$. During the sweeping process, two types of events (following [7,14]), *site event* and *circle event*, need to be handled.

(1) **Site event** (see Fig. 2): A site event occurs when the sweepline encounters an endpoint u of an input segment s. If u is the left endpoint of s, this event introduces a degenerate bisector, which is a horizontal ray, into the beachline. When the sweepline moves right, this degenerate bisector grows into a full k-segment bisector for the subsegment of s to the left of the sweepline. If u is the right endpoint of s, the event only updates the bisector of s. No new bisector is introduced.

(2) **Circle event** (see Fig. 3): A circle event occurs when two adjacent breakpoints of the beachline overlap (due to the growth of the adjacent bisectors). At this moment, three bisector segments meet at a common point (v as in Fig. 3) which forms a new breakpoint for the beachline. (Note that if more than three segments meet at the same point, the degenerate case can be easily handled by using a post-processing technique given in [8].) A circle event generates a new breakpoint and the bisector segments which are no longer in the beachline will be removed.

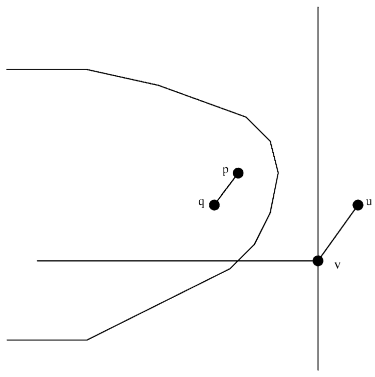

Fig. 2. Site event.

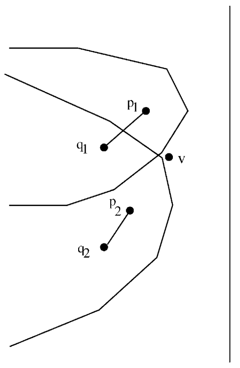

Fig. 3. Circle event.

3. Plane sweep algorithm for k-gon Voronoi diagram

Our plane sweep algorithm uses two main data structures to keep track of the *beachline* and the *event list*. Since the beachline is y-monotone, its vertices are maintained by a balanced binary search tree T (using their y-coordinates as the keys). The event list is stored in a priority queue, called *event queue* Q. The priority of each event is the time (i.e., the x-coordinate of L) when the event occurs. For a site event, the time is simply the x-coordinate of the endpoint of the corresponding line segment. For a circle event, the time is the x-coordinate of L when the two breakpoints overlap. Along with a circle event, we also store in Q the three bisector segments which cause the event. The main steps of the plane sweeping algorithm are the follows.

(1) **Initialization.** Initialize the data structures, i.e., a balanced binary search tree T for the beachline, a priority queue Q for the event queue. For each input segment, compute its ch-index based on its slope, and insert two site events into the event queue Q using the x-coordinates of the two endpoints as the keys.
(2) **Sweeping.** Repeatedly extract an event from the event queue Q until it is empty. For a site event, first insert a degenerate bisector into the beachline using the y-coordinate of the site event as the key. Then compute two new potential circle events for the newly added breakpoint and its two neighboring breakpoints, and insert them into the event queue. For a circle event, finalize the breakpoint as a Voronoi vertex, remove the vanished bisector segment from the beachline, compute circle events for the newly introduced breakpoint and its neighboring breakpoints in the beachline, and insert the circle events into event queue Q.

Notice that during the sweep, *false-alarm* problem may exist. For example, when we update the beachline, three consecutive bisector segments may meet at a common point at some time t, thus a potential circle event is generated and inserted into the event queue. However, should one of the three segments be removed from the beachline before time t (due to some other circle events), the circle event is not going to happen and must be removed from the event queue. Such a problem can be easily handled by checking for each encountered circle event whether the three involving bisector segments are still in the beachline before actually processing it.

The algorithm runs in $O(nk\log(nk))$ time and $O(nk)$ space for n input segments. The reason is as follows. Each bisector has $O(k)$ bisector segments and there are at most $O(nk)$ breakpoints in the beachline. Thus, an insertion or deletion operation in the tree T takes $O(\log(nk))$ time. For the event queue, there are $O(n)$ site events, and at most $O(nk)$ circle events. Thus the running time of each priority queue operation takes $O(\log nk)$ time. The major cost for each event is to perform $O(1)$ operations in T and Q and thus can be bounded by $O(\log(nk))$ time. The total time of the algorithm is therefore $O(nk\log(nk))$.

4. Degree analysis

A nice property of the above plane sweep algorithm is its low degree under certain k-gon metrics. To analyze the degree, we consider an arbitrary input segment $\overline{pq}$: $\overline{(x_p, y_p)(x_q, y_q)}$ with $x_p > x_q$ and $y_p > y_q$ (other cases can be similarly handled). As demonstrated in previous section, there are two types of bisector segments in the bisector between an input segment and the sweepline L: $x = t$ (see Fig. 1). One bisector segment is the segment connecting the pair of ch-index vertices of the bisectors of p and q. Others are segments on the bisector of p or q. Below are their equations:

$$y = A_i(x - x_g - d_g B_i) + y_g + d_g C_i, \tag{1}$$

$$D_j(x - x_q) - d_x(y - y_q) + d_q(d_x \sin j\theta - d_y \cos j\theta) = 0, \tag{2}$$

where $g \in \{p, q\}$.

In the above equations, all coefficients are functions of $\sin(i\theta)$ and $\cos(i\theta)$, $0 \leqslant i < k$, which are in general radicals. This seemingly suggests that the degree of the computation on these equations is high. To accurately analyze the degree of the algorithm, we consider the case of $k = 8$. Observe that under the octagon metric, although all the coefficients are still radicals, each of them is a product of a rational number and a common radical number $\sqrt{2}$. To make use of this nice property, we consider a special type of radicals (called β numbers) $\beta = (a + b\sqrt{s})/(c + d\sqrt{s})$, where a, b, c, d and s are some small rational numbers. The following lemma can be easily proved.

Lemma 1. *Let β be defined as above. Then, β numbers are closed under operations of addition, subtraction, multiplication, and division.*

To study the robustness issue of β numbers, we follow the degree model introduced by Liotta et al. [11]. In this model, the input variables are of degree 1 and denoted by α. A specific term ρ of degree s over input variables can be rewritten as α^s (genericization rule [11]). To compute the degree of unspecified polynomials, the following rules [11] were introduced:

$\alpha^s\alpha^r \to \alpha^{r+s}$	(product rule)
$\alpha^s + \alpha^s \to \alpha^s$	(sum rule)
$-\alpha^s \to \alpha^s$	(sign rule)
$\frac{\rho_j}{\rho_i} \pm \frac{\rho_h}{\rho_i} \to \rho_j \pm \rho_h$	(common denominator)
$\frac{\rho_j}{\rho_i} \pm \frac{\rho_h}{\rho_k} \to \rho_j\rho_k \pm \rho_i\rho_h$	(common denominator)
$\rho_i \pm \rho_j \to \rho_i^2 - \rho_j^2$	(segregate and square)

The degree of an algorithm is the highest degree of any test computation (or predicate), which is the evaluation of multivariate polynomial. For more information about the degree model and exact computation, readers are referred to [11,4,5].

To study the degree related to β numbers, we extend the degree model [11] with the following simple rewriting rules in the follow lemma, where α represents input variables of degree 1.

Lemma 2.

(1) $\beta \pm \beta \to \beta$,
(2) $\beta\beta \to \beta$,
(3) $\beta^{-1} \to \beta$,
(4) $\beta\alpha^n \to \alpha^{2n}$,
(5) $\beta\alpha^m \pm \beta\alpha^n \to \beta\alpha^{\max\{m,n\}}$.

Proof. These rewriting rules can be proved by using the rewriting rules given in [11]. Below we give the proof for Rule (4). Others are simpler and can be proved similarly.

By definition, $\beta\alpha^n = \frac{a+b\sqrt{s}}{c+d\sqrt{s}}\alpha^n$, and a, b, c, d, s are rational numbers. Using the rewriting rules in [11], we have the following.

$$\begin{aligned}\frac{a+b\sqrt{s}}{c+d\sqrt{s}}\alpha^n &\to \frac{(a+b\sqrt{s})(c-d\sqrt{s})}{c^2-d^2s}\alpha^n\\ &\to \frac{(ac-bds)+(bc-ad)\sqrt{s}}{c^2-d^2s}\alpha^n\\ &\to \alpha^n + \sqrt{s}\alpha^n\\ &\to \alpha^{2n} - s\alpha^{2n}\\ &\to \alpha^{2n}.\end{aligned}$$

In the rewriting process, we first multiply $c - d\sqrt{s}$ to both the numerator and the denominator, then simplify it by separating the term with irrational numbers and the one without, i.e., $\alpha^n + \sqrt{s}\alpha^n$. The small rational number a, b, c, d, s are constants and can be removed from the degree calculation. Using the segregate and square rule, we can eliminate the square root factor and rewrite $\beta\alpha^n$ as α^{2n}. □

Note that for β numbers, we only have one common radical number (i.e., $\sqrt{2}$ in octagon metric) which can be computed in an off-line fashion (and thus treated as a constant) to a precision which ensures the correctness of the exact computation. For example, if the desired precision is b bits, and the degree of the algorithm is d, then the radical number will have a precision with $d \times b + O(1)$ bits.

In our algorithm, the coordinates x_g, y_g of point g are input variables with degree 1. Using the rewriting rules, we can first rewrite A_i, B_i, C_i, D_i in Eqs. (1) and (2) as $A_i \to \beta$, $B_i \to \beta$, $C_i \to \beta$, $D_i \to \alpha\beta$. This is because all the $\sin(i\theta)$ and $\cos(i\theta)$ are multiple of the same radical $\sqrt{2}$ under octagon metric. In general, if we use other k-gon metric with the same property (i.e., only one radical/β number is involved in the computation of all the $\sin(i\theta)$ and $\cos(i\theta)$), the degree of the algorithm will be the same as in the case of octagon metric, e.g., $k = 6, 12$.

In our plane sweep algorithm, besides the computation of the ch-index, all the other test computations (predicates) occur when we are trying to insert an event into the event queue or insert a new (degenerate) bisector into the beachline. The comparison is either between the x-coordinate or y-coordinate of an endpoint of an input segment and a breakpoint or between the time of two circle events. In the following lemma, we identify 4 types of predicates and calculate their degrees.

Lemma 3. *In the plane sweeping algorithm, the following* 4 *types of predicates need to be computed exactly to ensure the accuracy of the algorithm, and the highest degree among them is* 14 *under the octagon metric.*

(A) $slope_s < \tan(i\pi/k)$ (*for comparing the slope of an input segment* s *with the slope of an edge of the regular polygon to determine the* ch*-index*).
(B) $y_p < y_{bp}$ (*for comparing the y-coordinate of a point* p *with a breakpoint* bp *in the beachline to insert a new bisector* (*i.e., a horizontal ray*) *generated from a site event at* p).
(C) $x_p < t_e$ (*for comparing the time of a site event which is the* x*-coordinate of a point* p *with the time of a circle event to insert the circle event* e *into the event queue*).
(D) $t_e < t_{e'}$ (*for comparing the times of two circle events* e *and* e' *to insert the new circle event* e *into the event queue*).

Proof. The existence of the four types of predicates follows directly from previous discussion on the sweeping algorithm. The time of a site event is the x-coordinate of an endpoint p of an input segment. A circle event occurs when three consecutive bisector segments in the beachline meet at a common point.

As for the degree, Predicate (A) has degree 2, since

$$\begin{aligned} slope_s - \tan(i\pi/k) &\to \frac{dy}{dx} - \beta \to \frac{\alpha}{\alpha} - \beta \\ &\to \alpha - \alpha\beta \to \alpha^2 - \alpha^2 \\ &\to \alpha^2. \end{aligned}$$

For Predicate (B), its degree is the highest when the breakpoint is the common intersection of three bisector segments (i.e., a breakpoint corresponding to a circle event). For Predicates (C) and (D), their degrees depend on the computation of the times associated with the circle events e and e'.

To determine the degrees of Predicates (B), (C) and (D), we consider a circle event generated by three bisector segments

$$a_i x + b_i y + c_i t = d_i,$$

$$a_j x + b_j y + c_j t = d_j, \quad \text{and}$$

$$a_k x + b_k y + c_k t = d_k.$$

Solving this linear system, we have

$$x = \begin{vmatrix} b_i & b_j & b_k \\ c_i & c_j & c_k \\ d_i & d_j & d_k \end{vmatrix} \Bigg/ \begin{vmatrix} a_i & a_j & a_k \\ b_i & b_j & b_k \\ c_i & c_j & c_k \end{vmatrix},$$

$$y = -\begin{vmatrix} a_i & a_j & a_k \\ c_i & c_j & c_k \\ d_i & d_j & d_k \end{vmatrix} \Bigg/ \begin{vmatrix} a_i & a_j & a_k \\ b_i & b_j & b_k \\ c_i & c_j & c_k \end{vmatrix},$$

$$t = \begin{vmatrix} a_i & a_j & a_k \\ b_i & b_j & b_k \\ d_i & d_j & d_k \end{vmatrix} \Bigg/ \begin{vmatrix} a_i & a_j & a_k \\ b_i & b_j & b_k \\ c_i & c_j & c_k \end{vmatrix}.$$

Using the rewriting rules, we first rewrite A_i, B_i, C_i, D_i as $A_i \to \beta$, $B_i \to \beta$, $C_i \to \beta$, $D_i \to \alpha\beta$, and then rewrite Eqs. (1) and (2) as:

$$\beta x + y + \beta t = \alpha\beta, \tag{3}$$

$$\alpha\beta x + \alpha y + \alpha\beta t = \alpha^2\beta. \tag{4}$$

It is clear that the degree of computing the intersection point of three bisector segments of the form of Eq. (4) is higher. And in this case, we have

$$x = \frac{\alpha^4\beta}{\alpha^3\beta}, \qquad y = \frac{\alpha^4\beta}{\alpha^3\beta}, \quad \text{and} \quad t = \frac{\alpha^4\beta}{\alpha^3\beta}.$$

Plugging them into Predicates (B), (C), and (D), we have

$$\text{(B)} \quad y_p - \frac{\alpha^4\beta}{\alpha^3\beta} \to y_p\alpha^3\beta - \alpha^4\beta \to \alpha^4\beta \to \alpha^8,$$

$$\text{(C)} \quad x_p - \frac{\alpha^4\beta}{\alpha^3\beta} \to \alpha^4\beta \to \alpha^8,$$

$$\text{(D)} \quad \frac{\alpha^4\beta}{\alpha^3\beta} - \frac{\alpha^4\beta}{\alpha^3\beta} \to \alpha^7\beta \to \alpha^{14}. \qquad \square$$

In VLSI applications, the input segments are often rectilinear or oriented with slope of $\pi/4$ or $3\pi/4$ [13]. In the cases of $\pi/4$ and $3\pi/4$, since the slope of input segments is $dy/dx = 1$, Eq. (4) can be further simplified to Eq. (3). And the rectilinear case is even simpler. Thus the intersection of three bisector segments of Eq. (3) will be

$$x = \frac{\alpha\beta}{\beta}, \qquad y = \frac{\alpha\beta}{\beta}, \qquad t = \frac{\alpha\beta}{\beta},$$

and the degree of Predicate (D) is

$$\frac{\alpha\beta}{\beta} - \frac{\alpha\beta}{\beta} \to \alpha\beta \to \alpha^2.$$

It is interesting to point out that the incircle test under the k-gon metric have predicates (which need to be computed exactly) similar to above ones and therefore has the same degree.

Lemma 4. *For certain k-gon metrics* (*e.g.*, $k = 6, 8, 12$), *the incircle test for disjoint line segments has degree at most* 14.

Theorem 5. *For a set of non-crossing segments with arbitrary orientation, the plane-sweeping algorithm for computing the k-gon* (*for* $k = 6, 8, 12, \ldots$) *Voronoi diagram has degree at most* 14. *If the orientation of the segments is either rectilinear, $\pi/4$, or $3\pi/4$, then the degree is reduced to* 2.

Acknowledgements

The authors would like to thank Professor Francis Y.L. Chin and two anonymous referees for their thoughtful comments which helped to improve the presentation of this paper.

References

[1] F. Aurenhammer, Voronoi diagrams—a survey of a fundamental geometric data structure, ACM Comput. Surv. 23 (1991) 345–405.

[2] F. Aurenhammer, R. Klein, Voronoi diagrams, in: J. Sack, G. Urrutia (Eds.), Handbook of Computational Geometry, Elsevier Science Pub., Amsterdam, 2000, pp. 201–290.

[3] M. de Berg, M. van Kreveld, M. Overmars, O. Schwarzkopf, Computational Geometry: Algorithms and Applications, Springer-Verlag, Berlin, 2000.

[4] C. Burnikel, Exact computation of Voronoi diagrams and line segment intersections, Ph.D. Dissertation, Universität des Saarlandes, Saarbrücken, Germany, 1996.

[5] C. Burnikel, K. Mehlhorn, S. Schirra, How to compute the Voronoi diagram of line segments: theoretical and experimental results, in: Proc. ESA, in: Lecture Notes in Comput. Sci., vol. 855, Springer, Berlin, 1994, pp. 227–239.

[6] L.P. Chew, R.L. Drysdale, Voronoi diagrams based on convex distance functions, in: Proc. ACM Symp. Comp. Geom., 1985, pp. 235–244.

[7] F. Dehne, R. Klein, The big sweep: on the power of the wavefront approach to Voronoi diagrams, Algorithmica 17 (1997) 19–32.

[8] S.J. Fortune, A sweepline algorithm for Voronoi diagrams, Algorithmica 2 (1987) 153–174.

[9] V. Karamcheti, C. Li, I. Pechtchanski, C. Yap, A core library for robust numeric and geometric computation, in: Proc. ACM Symp. on Computational Geometry, 1999.

[10] R. Klein, Concrete and Abstract Voronoi Diagrams, Lecture Notes in Comput. Sci., vol. 400, Springer-Verlag, Berlin, 1989.

[11] G. Liotta, F.P. Preparata, R. Tamassia, Robust proximity queries: an illustration of degree-driven algorithm design, SIAM J. Comput. 28 (3) (1998) 864–889.

[12] M. McAllister, D. Kirkpatrick, J. Snoeyink, A compact piecewise-linear Voronoi diagram for convex sites in the plane, Discrete Comput. Geom. 15 (1996) 73–105.

[13] E. Papadopoulou, D.T. Lee, Critical area computation via Voronoi diagrams, IEEE Trans. Comput.-Aided Design 18 (1999) 463–474.

[14] E. Papadopoulou, D.T. Lee, The L_∞ Voronoi diagram of segments and VLSI applications, Internat. J. Comput. Geom. Appl. 11 (5) (2001) 503–528.

[15] F.P. Preparata, M.I. Shamos, Computational Geometry—An Introduction, Springer-Verlag, New York, 1985.

[16] M. Sarrafzadeh, C.K. Wong, Hierarchical Steiner tree construction in uniform orientation, IEEE Trans. Comput.-Aided Design 11 (8) (1992) 1095–1103.

[17] P. Widmayer, Y.F. Wu, C.K. Wong, On some distance problems in fixed orientations, SIAM J. Comput. 16 (1987) 728–746.

[18] C. Yap, Towards exact geometric computation, Comput. Geom. 7 (1) (1997) 3–23.