International Journal of Computational Geometry
& Applications
Vol. 25, No. 2 (2015) 123–141

DOI: 10.1142/S0218195915500089

World Scientific
www.worldscientific.com

The L_∞ Hausdorff Voronoi Diagram Revisited*

Evanthia Papadopoulou†
Faculty of Informatics, USI - Università della Svizzera italiana
Lugano, Switzerland
evanthia.papadopoulou@usi.ch

Jinhui Xu
Department of Computer Science and Engineering
State University of New York at Buffalo, USA
jinhui@buffalo.edu

Received 1 April 2014
Revised 5 February 2015
Published 20 July 2015
Communicated by K. Sugihara

ABSTRACT

We revisit the L_∞ Hausdorff Voronoi diagram of clusters of points in the plane and present a simple two-pass plane sweep algorithm to construct it. This problem is motivated by applications in the semiconductor industry, in particular, *critical area analysis* and *yield prediction* in VLSI design. We show that the structural complexity of this diagram is $\Theta(n + m)$, where n is the number of given clusters and m is a number of specially *crossing* clusters, called *essential*. Our algorithm runs in $O((n + m') \log n)$ time and $O(n + m')$ space, where m' reflects a slight superset of essential crossings, $m \leq m' \leq M$, and M is the total number of crossing clusters. For non-crossing clusters ($M = 0$) or clusters with only a small number of crossings ($M \in O(n)$) the algorithm is optimal. The latter is the case of interest in the motivating application, where $M \ll n^2$. This is achieved by augmenting the *wavefront* data structure of the plane sweep, and a preprocessing step, based on *point dominance*, which is interesting in its own right.

Keywords: Voronoi diagram; Hausdorff distance; plane sweep; L_∞ metric; point dominance; VLSI layout.

*A preliminary version of this paper has appeared in Ref. 22.
†Research supported in part by the Swiss National Science Foundation, projects 200021-127137, 200020-149658, and the ESF EUROCORES program EuroGIGA/VORONOI, SNF 20GG21-134355.

1. Introduction

Given a set S of point clusters in the plane, the *Hausdorff Voronoi diagram* of S, denoted $HVD(S)$, is a subdivision of the plane into regions such that the *Hausdorff Voronoi region* of a cluster P is the locus of points *closer* to P than to any other cluster in S. The distance between a point t and a cluster P is the *farthest distance* between t and all points in P, $d_f(t, P) = \max\{d(t, p), p \in P\}$, where $d(t, p)$ denotes the ordinary distance between points t, p. The farthest distance $d_f(t, P)$ is equivalent to the *Hausdorff distance*[a] between t and P, hence, the name of the diagram. The Hausdorff Voronoi region of P is

$$hreg(P) = \{x \mid d_f(x, P) < d_f(x, Q), \forall Q \in S\}$$

and it is subdivided into finer regions by the *farthest Voronoi diagram* of P, $FVD(P)$.

In the L_∞ metric,[b] $d_f(t, P) = d_f(t, P')$, where P' is the minimum enclosing (axis-aligned) rectangle of P. As a result, the L_∞ Hausdorff Voronoi diagram of a set S of n clusters of points is equivalent to the L_∞ Hausdorff Voronoi diagram of their minimum enclosing rectangles. Once the minimum enclosing rectangles are available, individual cluster points have no further influence, and they can be ignored. In this paper, we use the terms cluster and rectangle interchangeably. In L_∞, two clusters are called *crossing* if their minimum enclosing rectangles intersect in the shape of a cross.

The Hausdorff Voronoi diagram has appeared in the literature under different names having been motivated by different problems.[1,4,7,8,16,17,25] It first appeared as the *cluster Voronoi diagram*,[8] where a powerful geometric transformation in three dimensions revealed structural properties and an $O(n^2\alpha(n))$ divide-and-conquer algorithm for its construction. In L_2, n is the number of points on the cluster convex hulls. The sructural complexity of the diagram was shown to be $O(n^2)$,[17] which also improved the complexity bound of the divide-and-conquer algorithm to $O(n^2)$. The L_∞ version of the problem was introduced as a solution to the VLSI *critical area extraction* problem for a defect mechanism, called a *via-block*. The L_∞ Hausdorff Voronoi diagram was shown equivalent to an L_∞ Voronoi diagram of additively weighted line-segments.[16] This resulted in a plane sweep algorithm, based on Fortune's plane sweep,[10] augmented with new events to handle the special features of the Hausdorff Voronoi regions.[16,17] The time complexity depended on a parameter K, where K was $O(n^2)$, even in the case of non-crossing rectangles when the complexity of the diagram is $O(n)$. Bounding the time complexity of the plane sweep construction, while maintaining its simplicity, had since remained an open problem. Note that the Voronoi diagram of additively weighted segments, is not an

[a]The (directed) Hausdorff distance from a set A to a set B is $h(A, B) = \max_{a \in A} \min_{b \in B}\{d(a, b)\}$. The (undirected) Hausdorff distance between A and B is $d_h(A, B) = \max\{h(A, B), h(B, A)\}$.

[b]The L_∞ distance between two points $p = (x_p, y_p)$, $q = (x_q, y_q)$ is $d(p, q) = \max\{|x_p - x_q|, |y_x - y_p|\}$.

instance of abstract Voronoi diagrams,[11] unlike its counterpart for points, thus, an efficient plane sweep adaptation is not an easy task. An $O(n \log n)$ claim[24] for a plane sweep construction of the Hausdorff Voronoi diagram for disjoint clusters in L_2 is erroneous.

In the Euclidean metric and for non-crossing clusters, there have been recent advances in the efficient construction of this diagram using the randomized incremental paradigm, however, no optimal algorithm has been available yet. In particular, for non-crossing clusters, the diagram can be computed in expected $O(n \log^2 n)$ time and expected $O(n)$ space.[4] It can be constructed in parallel in $O(p^{-1} n \log^4 n)$ time and p processors, which results in a divide and conquer sequential algorithm of $O(n \log^4 n)$ time and $O(n \log^2 n)$ space complexity.[7] The Hausdorff Voronoi diagram is a *min-max* type of diagram, thus, it is related to a *max-min* counterpart termed the *farthest color Voronoi diagram*.[2,5,9] For more information on Voronoi diagrams, see the book of Aurenhammer *et al.*[3] or Okabe *et al.*[15]

In this paper we revisit the plane sweep construction of the Hausdorff Voronoi diagram in the L_∞ metric and bound its complexity, resulting in an optimal algorithm for clusters with a small number of crossings. We first refine the structural complexity of the L_∞ Hausdorff diagram and show that it is $\Theta(n + m)$, where m, $0 \leq m \leq M$, is a number of specially crossing clusters, called *essential* (see Def. 1), and M is the total number of crossings. The algorithm runs in $O((n + m') \log n)$ time and $O(n + m')$ space, where m' reflects a superset of essential crossings, $0 \leq m \leq m' \leq M$ (see Def. 2). This is optimal for $m' \in O(n)$. We achieve the time complexity bound by two additions to the original plane sweep:[16] (1) a first pass plane-sweep preprocessing step, based on *point dominance* in $\mathbb{R}^3$; and (2) augmenting the standard *wavefront* data structure. The method directly extends to clusters of axis-parallel line-segments. This is a companion paper to Ref. 25, which presents an output sensitive version of the L_∞ plane sweep at the expense of increased space complexity. The output sensitive time complexity is not optimal for non-crossing clusters. This paper fills this gap with a two-pass plane sweep, which is optimal for clusters with a small number of crossings ($M \in O(n)$) and simple for use in the motivating application, where typically $0 < M \ll n^2$.

We remark that the L_∞ Hausdorff Voronoi diagram falls under the umbrella of *abstract Voronoi diagrams*[11,12] strictly for sets of non-crossing clusters ($M = 0$). In the presence of crossings ($M > 0$), even if only a small number is relevant, bisectors can be disconnected, and thus, the $O(n \log n)$-expected time algorithm for abstract Voronoi diagrams[13] is not applicable.

The L_∞ Hausdorff Voronoi diagram finds direct applications in VLSI design for manufacturability.[16,18,26] It provides a solution to the *geometric min-cut problem*,[18] which addresses the critical area extraction problem for open faults in VLSI designs in the presence of redundant interconnects. Due to the simplicity of the plain sweep paradigm and the absence of numerical issues in L_∞,[20] the approach is very well suited for practical implementation.

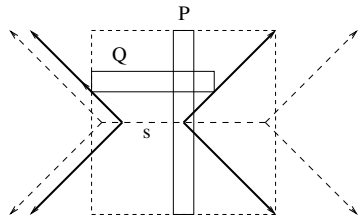

Fig. 1. The L_∞ Hausdorff bisector of crossing rectangles superimposed on $FVD(P)$.

This paper is organized as follows. Section 2 provides definitions and a tight bound on the structural complexity of the L_∞ Hausdorff Voronoi diagram. Section 3 gives a preprocessing step (Phase 1) based on *point dominance* in $\mathbb{R}^3$. Section 4 revisits the plane sweep construction, augments the standard *wavefront* data structure, and using the results of Phase 1, achieves the time complexity bound. Section 5 extends to clusters of axis-parallel line segments, and Sec. 6 concludes.

2. Definitions and Structural Complexity

Let S be a set of n rectangles, or equivalently, a set of n point clusters in the plane such that each cluster has been substituted by its minimum enclosing (axis-aligned) rectangle. A pair of rectangles (P, Q) is called *crossing* if P and Q intersect in the shape of a cross, i.e., $P \setminus Q$ is disconnected. Given a crossing pair (P, Q), the first rectangle P is assumed to be at least as long as Q. Given a rectangle P, let P^n, P^s, P^e, and P^w denote the north, south, east, and west edge of P, respectively. P is called horizontal (resp., vertical) if P^n is longer (resp., shorter) then P^e. The axis parallel line through edge P^i, $i = n, s, e, w$, is denoted as $l(P^i)$. The term P^i is also used to denote the axis-parallel coordinate of edge P^i.

Let the *core segment* of P be the locus of centers of all minimum enclosing squares of P. It corresponds to the axis-parallel line segment of the L_∞ *farthest Voronoi diagram* of P ($FVD(P)$), which degenerates to a point in case P is a square. It can be viewed as an ordinary line segment s additively weighted with $w(s) = d_f(s, P)$. In Fig. 1, $FVD(P)$ is illustrated in dashed lines and the core segment of P is indicated by s. $FVD(P)$ consists of exactly four regions delimited by s and four rays of slope ± 1 emanating from the endpoints of s. The L_∞ Hausdorff Voronoi diagram of S is equivalent to the (weighted) Voronoi diagram of the set of core segments of all clusters in S.[16]

The Hausdorff bisector between two clusters P, Q is $b_h(P, Q) = \{y \mid d_f(y, P) = d_f(y, Q)\}$ and it forms a subgraph of $FVD(P \cup Q)$.[21] For non-crossing rectangles, $b_h(P, Q)$ is a single unbounded chain consisting of at most three segments as derived from $FVD(P \cup Q)$ (see, e.g., Fig. 2). For crossing rectangles, $b_h(P, Q)$ consists of exactly two unbounded chains, as formed by the four ± 1-slope rays of $FVD(P \cup Q)$ (see, e.g., Fig. 1). In the presence of collinear edges between P and Q, entire regions of $FVD(P \cup Q)$ may be equidistant from both P and Q (see, e.g., Fig. 2(a)).

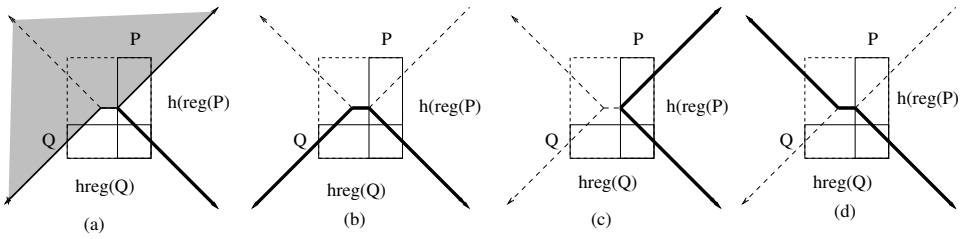

Fig. 2. The L_∞ Hausdorff bisector of rectangles P and Q in the presence of collinear edges. (a) Shaded regions are equidistant from both P and Q. (b) Equidistant regions are assigned to P. (c) Equidistant regions are assigned to Q. (d) Equidistant regions are assigned one to P and one to Q.

Equidistant regions are typically assigned to only one of the sites, thus, the exact shape of $b_h(P,Q)$ (shown bold in Fig. 2) depends on the conventions to break ties. For the sake of simplicity, collinear edges are always treated as non-crossing configurations.

For a rectangle Q strictly enclosed in the interior of a minimum enclosing square of P, $b_h(P,Q)$ consists of either one (if P and Q are non-crossing) or two (if P and Q are crossing) chains, each one forming a V-shape derived by the ± 1-slope rays of $FVD(P \cup Q)$. The apex of each chain is called a *V-vertex*. A V-vertex v is always incident to the core segment of P and its 90°-angle faces the portion of the plane that is closer to P than Q. A V-vertex is characterized as *up*, *down, right*, or *left*, depending on whether its 90°-angle is facing north, south, east, or west respectively. In addition, it is characterized as *crossing*, if Q is crossing P, and *non-crossing*, otherwise. The minimum enclosing square of P centered at v is denoted as $square(P,v)$ and it must be enclosing Q. It is also denoted as $square(P,Q^i)$, where Q^i, is the non-crossing edge of Q that delimits one of its edges. Figure 1 illustrates $b_h(P,Q)$ consisting of two crossing V-vertices, one right and one left; $square(P,Q^w)$ is illustrated dashed. $square(P,Q^i)$ is referred to as an *extremal minimum enclosing square* of P and Q.

The V-vertices of $HVD(S)$ are called *Voronoi V-vertices*, but they have also been called *mixed Voronoi vertices*.[17] Recall that the Hausdorff Voronoi diagram has two types of vertices: (1) ordinary Voronoi vertices, each one equidistant from three different clusters; and (2) mixed vertices, each one equidistant from two different clusters, corresponding to a *breakpoint* of a Hausdorff bisector. The structural complexity of the diagram is proportional to the number of its mixed Voronoi vertices, and the number of non-crossing mixed vertices is $O(n)$.[17,21] Thus, the complexity of the L_∞ Hausdorff Voronoi diagram is $\Theta(n+m)$, where m is the number of its crossing Voronoi V-vertices.

Definition 1. A pair of crossing rectangles (P,Q) is called *essential* if there is an extremal minimum enclosing square of P and Q, $square(P,Q^i)$, that is empty of other rectangles in S.

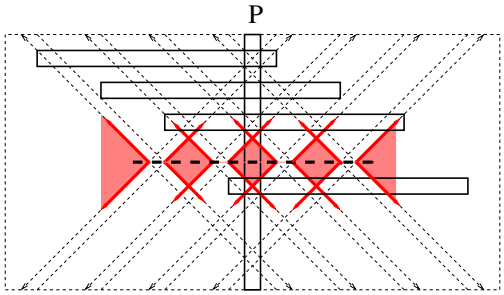

Fig. 3. Collection of essential crossings. Shaded regions belong to P.

Lemma 1. *A pair of crossing rectangles (P, Q) induces a Voronoi V-vertex in $HVD(S)$ if and only if (P, Q) is an essential crossing.*

Proof. Without loss of generality let P be a vertical rectangle. Then $b_h(P, Q)$ consists of exactly two V-vertices, v_w and v_e, corresponding to the centers of $square(P, Q^w)$ and $square(P, Q^e)$ respectively. By definition of the Hausdorff Voronoi diagram, v_w (resp., v_e) remains a Voronoi V-vertex in $HVD(S)$ if and only if $square(P, Q^w)$ (resp., $square(P, Q^e)$) encloses no other rectangle. Since no other V-vertex can be induced by $b_h(P, Q)$, the claim follows. □

By Lemma 1 and structural complexity results in L_2,[17,21] we conclude.

Theorem 1. *The structural complexity of the L_∞ Hausdorff Voronoi diagram of a set S of n point clusters, equivalently n rectangles, is $\Theta(n + m)$, where m is the total number of crossings that are essential.*

Definition 2. Let P be a vertical rectangle. A collection of crossings (P, Q_i), $i = 1, \ldots, k$, is called a *staircase*, if $Q_i^w < Q_{i+1}^w$ and $Q_i^e < Q_{i+1}^e$, $i = 1, \ldots, k$, and for every Q_i, $square(P, Q_i^w)$ is empty of $Q_j \neq Q_i$. The maximum size of such a staircase for P is the maximum number of *potentially essential crossings* for P. Let m' denote the number of potentially essential crossings for all vertical rectangles plus the number of essential crossings for all horizontal rectangles in S.

Figure 3 shows a staircase for a vertical rectangle P consisting of horizontal rectangles $Q_1, \ldots, Q_k$. Considering $S = \{P, Q_i, 1 \leq i \leq k\}$, the Voronoi region of P is shown shaded and it is disconnected, periodically alternating with regions of crossing rectangles. In the absence of additional rectangles, all crossings are essential, i.e., they all induce Voronoi V-vertices in the Hausdorff Voronoi diagram of S. The staircase $Q_1, \ldots, Q_k$ can be constructed arbitrarily large as follows: Let $\varepsilon > 0$ be a small constant. Let $Q_i, 1 \leq i \leq k$, be a collection of horizontal rectangles, crossing P, each one of length equal to the length of P minus $\varepsilon/2$, such that Q_1^e is located ε to the right of P^e, and Q_i^e is located ε to the right of Q_{i-1}^e, for any $i, 1 \leq i \leq k$. Considering $S = \{P, Q_i, 1 \leq i \leq k\}$, the region of P in $HVD(S)$

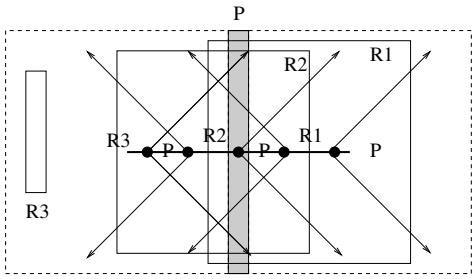

Fig. 4. The dominating sequence of rectangle P consisting of rectangles R_1, R_2, R_3.

periodically alternates with regions of $Q_i, 1 \leq i \leq k$, in increasing order, starting with a subregion of P (see Fig. 3). The V-vertices of all bisectors $b(P, Q_i)$ remain Voronoi V-vertices in $HVD(S)$.

Our algorithm consists of two independent plane-sweep phases. Phase 1 is a preprocessing step that collects information on *dominating sequences* of vertical rectangles (defined below) to speed up Phase 2. The construction of the diagram is performed in Phase 2, using the information obtained in Phase 1 to maintain its time complexity. In what follows we define these concepts. Throughout the paper we assume a vertical sweep-line.

Definition 3. The *dominating-sequence* of a vertical rectangle P, denoted $ds(P)$, is a maximal collection of vertical rectangles $R_i, i = 1, \ldots, k$, such that every R_i is entirely enclosed in some minimum enclosing square of P, R_1 is the rectangle with the rightmost west edge among all rectangles dominated by P, and if R_i is crossing P, $i \geq 1$, then R_{i+1} is the vertical rectangle with the rightmost west edge dominated by $square(P, R_i^e)$.

By definition, $ds(P)$ forms a staircase for P (except the last rectangle, if it is non-crossing with P). In the case of non-crossing rectangles, $ds(P) = (R_1)$, unless $ds(P) = \emptyset$. Figure 4 illustrates the dominating sequence of a vertical rectangle P consisting of three rectangles R_1, R_2, R_3. Considering only these four rectangles, the Hausdorff Voronoi region of P is alternating with regions of R_i on the core segment of P, as shown in Fig. 4.

Lemma 2. *No vertical rectangle, other than the dominating sequence of P, may induce a right Voronoi V-vertex on the core segment of P.*

Proof. Let Q be a vertical rectangle enclosed in some minimum enclosing square of P such that Q is not included in $ds(P)$. Recall that only rectangles enclosed in a minimum enclosing square of P may induce a V-vertex on the core segment of P. Let R_i be the rectangle in $ds(P)$ of maximum index such that $Q^w < R_i^w$ (i.e., $R_{i+1}^w < Q^w$, $i < k$). Since $Q \neq R_{i+1}$, Q^w must be enclosed in $square(P, R_i^e)$. But then the right branch of $b_h(P, Q)$ (defined by Q^w) must be in between the two

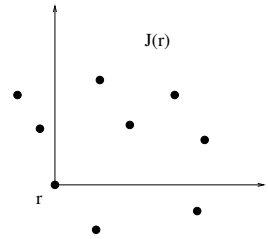

Fig. 5. The range $J(r)$ for a point r.

branches of $b_h(P, R_i)$, and thus, the right V-vertex of $b_h(P, Q)$ is subsumed within the region of R_i. □

3. Preprocessing Step – Point Dominance

Phase 1 computes the dominating sequence of vertical rectangles as based on *point dominance*.

For any vertical rectangle P, map P^w into point $p = (P^n, -P^s, P^w)$ in $\mathbb{R}^3$, denoted as $point(P)$. A point $p = (p_x, p_y, p_z)$ is said to *dominate* a point $q = (q_x, q_y, q_z)$ if $p_i \geq q_i$ for all $i = x, y, z$. Given a point p, let $R(p)$ denote the rectangle that is mapped into p. If two different rectangles map into the same point p, let $R(p)$ be the one with the leftmost east edge. Note that any rectangle, other than $R(p)$, that maps onto the same point p must have an empty Hausdorff Voronoi region, and thus, it can be discarded.

To determine the dominating sequence of every vertical rectangle P, we first determine R_1, the rectangle dominated by P with the rightmost west edge (if any). In terms of point dominance, this problem is equivalent to determining for every point p, the topmost point q dominated by p. This is a variant of the standard *point dominance* problem in R^3.[23] This variant can be solved in $O(n \log n)$ time and $O(n)$ space, where n is the number of points, by combining a simple sweeping algorithm and the use of an auxiliary *priority search tree*,[14] T. Every point p has an *expiration time* corresponding to the west edge of its leftmost minimum enclosing square, $square(P, P^e)^w$, after which p is permanently deleted from T.

In detail, we sweep points in decreasing z-coordinate while maintaining their *minima*, $M(t)$, where $M(t)$ is the set of points above the sweeping plane at $z = t$, which do not dominate anything so far, projected on the xy-plane. The set of minima $M(t)$ is organized in a priority search tree T. Let $r = (r_x, r_y, r_z)$ be the point to be considered at time $t = r_z$. Perform a range query on T for the NE quadrant of (r_x, r_y), i.e., range $J(r) = [r_x, \infty] \times [r_y, \infty]$, as shown in Fig. 5. For any point p within $J(r)$, report (p, r), assuming that the expiration time of p has not been reached yet, else simply discard p. Delete all points in $J(r)$ from T and insert r in T. Since priority search trees are fully dynamic, this is an $O(n \log n)$ time $O(n)$ space algorithm.

For non-crossing rectangles, or if R_1 is non-crossing with P, this completes the search for $ds(P)$. In the presence of crossings, however, $ds(P)$ may contain additional rectangles. Let R be a rectangle in $ds(P)$, crossing with P. Then, $point(P) \in J(R)$, thus, $point(P)$ is deleted from T when we consider $point(R)$. To be able to determine the rectangle in $ds(P)$ following R, a point representing P should be re-inserted in $M(t)$ at time $t = square(P, R^e)^w$. To achieve this, we consider a new rectangle $Q = square(P, R^e)$. In particular, let $point(Q) = (p_x, p_y, p'_z)$, where $p'_z = Q^w$; let $R((p_x, p_y, p'_z)) = P$; and let the expiration time of $point(Q)$ be the one of P, i.e., $square(P, P^e)^w$. We can treat Q as a normal rectangle such that $point(Q)$ is inserted in T at time $t = Q^w$.

To this aim, we maintain an additional max-heap H of new rectangles Q, where Q represents a pair of crossing rectangles (P, R), $Q = square(P, R^e)$, such that the expiration time of P has not been reached yet, and R is the last rectangle in $ds(P)$ discovered so far. H is organized according to the maximum west edge of its squares. When a square Q in H is processed, $point(Q)$ is simply inserted in T. At any time, the next rectangle to be processed is the one of maximum west edge among the original rectangles and the squares in H. Original rectangles are processed normally as described above.

The process repeats until either a rectangle non-crossing with P, dominated by P, is reached (the last rectangle in $ds(P)$) or the expiration time of P is reached, at which time P is permanently deleted from T. The algorithm is summarized in the Appendix. At the end of the sweep, $ds(P)$ is available for every vertical rectangle.

Correctness follows by the definition of a dominating-sequence. The time complexity is $O((n + d) \log n)$, where d is the size of dominating sequences of vertical rectangles. The consumed space is $O(n+d)$. By Lemma 2, d is $O(m')$, thus, the time complexity of Phase 1 is $O((n + m') \log n)$ and the space complexity is $O(n + m')$.

4. A Refined Plane Sweep Construction

In this section we revisit the plane sweep construction of the L_∞ Hausdorff Voronoi diagram,[16] generalize it to crossing rectangles, and improve its time complexity. The algorithm is based on the plane sweep paradigm for Voronoi diagrams,[6,10] its adaptation for line-segments in L_∞,[20] and its generalization to handle special features of Hausdorff Voronoi diagrams.[16,17] We use two new additions to achieve the bounded time complexity: (1) the dominating sequences of Phase 1; and (2) augmenting the main data structure of the plane sweep to efficiently answer a variety of queries.

4.1. *Overview*

Assume a vertical sweep-line l_t sweeping the plane from left to right. At any instant t of the sweeping process we compute $HVD(S_t \cup l_t)$, for $S_t = \{P \in S \mid l(P^e) < t\}$. The boundary of the Voronoi region of l_t is the *wavefront* at time t. Voronoi edges and core segments incident to the wavefront are called *spike bisectors* and *spike*

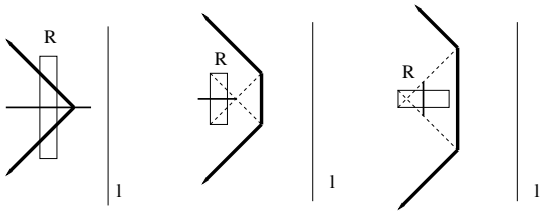

Fig. 6. The wavecurve of a rectangle R.

core segments respectively. The combinatorial structure of the wavefront changes at specific *events* organized in a priority queue. There are two types of events, *site events*, when new waves may appear on the wavefront, and *spike events*, when waves disappear due to the intersection of their incident spike bisectors. Spike events remain essentially the same as in the ordinary plane sweep and we skip their discussion in this paper. We have four types of *site events*: *start-vertical-rectangle*, *end-vertical-rectangle*, *V-vertex* events (for brevity *V-events*) for vertical rectangles, and *horizontal-rectangle* events. V-events are generated to predict right V-vertices along horizontal core segments. V-events corresponding to Voronoi vertices of the final diagram are called *valid*; else they are called *invalid*.

The *wavecurve* of a rectangle R is the bisector between R and the sweep line l_t, at time t, $b(R, l_t) = \{y \mid d_f(y, R) = d(y, l_t)\}$, where $d(y, l_t)$ is the ordinary distance between y and l_t. In L_∞, it consists of two or three *waves*: a ray of slope -1, corresponding to $b(R^s, l_t)$, a ray of slope $+1$, corresponding to $b(R^n, l_t)$, and possibly a vertical line-segment corresponding to $b(R^w, l_t)$. The wavecurve of R can be seen equivalently as the (weighted) bisector between the core segment s of R and l_t, i.e., $b(R, l_t) = \{y \mid d_w(y, s) = d(y, s) + w(s) = d(y, l_t)\}$. In Fig. 6, the wavecurve of several instances of rectangles is illustrated. The bold axis-aligned segment illustrates the core segment of R. The *wavefront*, at time t, is the lower envelope, with respect to the sweep-line, of the *wavecurves* of all rectangles in S_t. In L_∞, the wavefront is monotone with respect to any line of slope ± 1. The wavefront is typically maintained as a height balanced binary tree, $\mathcal{T}$, ordered from bottom to top. Leaf nodes correspond to waves, while internal nodes correspond to spike bisectors and spike core segments revealing *breakpoints* between incident waves. Furthermore, we augment the wavefront with additional values as explained in Sec. 4.2.

Any Voronoi point in $HVD(S)$ enters the wavefront at the time of its *priority*. The *priority* of a point v, *priority*(v), is the rightmost x-coordinate of the smallest square centered at v that is entirely enclosing the rectangle P that induces v. That is, *priority*(v) = $square(P, v)^e$. All events are processed at the time of their priority. The priority of a *start-rectangle* event is the x-coordinate of P^e, denoted as *priority*(P). The priority of an *end-rectangle* event is $square(P, P^w)^e$. The priority of a V-event v on core-segment s is $square(P, v)^e$. The priority of a start- and an

end-rectangle event is the minimum and the maximum priority respectively of the points on core segment s.

4.2. *Augmenting the wavefront*

To maintain efficiency, each node x of the wavefront is augmented with additional values as follows: (1) A *w-max* value representing the rightmost west edge of all rectangles contributing a wave to the portion of the wavefront rooted at x; and (2) Two *x-min* values (*x-min-I* and *x-min-II*) that in combination represent the minimum x-coordinate of the portion of the wavefront rooted at x. We use two *x-min* values instead of one so that updates of these values can take place exactly at regular events. In particular, *x-min-I* (resp. *x-min-II*) represents the breakpoint of minimum x-coordinate among the breakpoints incident to a horizontal (resp. slope ± 1) spike bisector in the portion of the wavefront rooted at x. In more detail, for a leaf node representing a wave of rectangle R, the *w-max* value is R^w and both *x-min* values are $+\infty$. For an internal node, *w-max* is the maximum between the *w-max* values of its children. For the same internal node, *x-min-I* (resp. *x-min-II*) points to the breakpoint of minimum x-coordinate among its own breakpoint and the *x-min-I* (resp. *x-min-II*) values of its children. The minimum x-coordinate of the portion of the wavefront rooted at node x is $x\text{-}min = \min\{x\text{-}min\text{-}I, x\text{-}min\text{-}II\}$.

The augmentation values *w-max, x-min-I, x-min-II* remain the same unless a combinatorial change in the wavefront (i.e., an event) takes place. This is the reason for keeping two separate *x-min* values instead of one, one for each type of spike bisector, to ensure that updates need only take place at the given events. For brevity, in the following, we assume only two augmentation values, *w-max* and *x-min*, where *x-min* is always available by combining *x-min-I* and *x-min-II*.

This augmentation of the standard wavefront data structure partially achieves the improved complexity of the main plane-sweep algorithm.

4.3. *Event handling*

Let us now discuss the handling of various types of site events for a rectangle P of core segment s (see Fig. 7). We mainly consider start-rectangle and V-vertex events, as end-rectangle events are similar to the original plane sweep,[16] and we skip their discussion.

At time t, let $r_1(t)$ and $r_2(t)$ be the rays of slope $+1$ and -1, respectively, emanating from $l(P^n) \cap l_t$, and $l(P^s) \cap l_t$ respectively, extending towards the left of l_t. Let $a(t)$ and $b(t)$ be the intersection points of $r_1(t)$ and $r_2(t)$ with the wavefront at time t, respectively. Since the wavefront is ± 1 monotone, $a(t)$ and $b(t)$ can be determined by binary search in $O(\log n)$ time. In case of a wave collinear with $r_1(t)$ or $r_2(t)$, the rightmost endpoint is assigned to $a(t), b(t)$, adopting the convention that an equidistant region is assigned to the rectangle preceding P. Because the wavefront is monotone with respect to any line of slope ± 1, in case of a vertical rectangle, the entire portion of the wavefront between $a(t)$ and $b(t)$ must be either

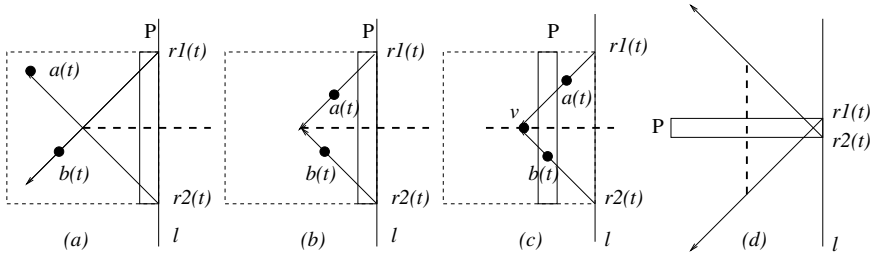

Fig. 7. (a) At $t = priority(P)$, the wavefront has not reached s. (b) At $t = priority(P)$, the wavefront has covered portion of s. (c) An invalid event at time $t = priority(v)$. (d) A horizontal rectangle event.

to the left (Fig. 7(a)) or the right (Figs. 7(b) and 7(c)) of the intersection point of $r_1(t)$ and $r_2(t)$; thus, it may intersect $r_1(t), r_2(t)$, or the core segment of s at most once. For a horizontal rectangle (Fig. 7(d)), the wavefront can intersect the vertical core segment of P an arbitrary number of times.

Consider time $t = priority(P)$, more generally, time $t = priority(v)$ for a site event v. There are three cases: (1) The wavefront has not reached core segment s yet (either at a start-vertical-rectangle or at a horizontal-rectangle event), or the wavefront touches a valid V-event, or the wavefront touches the endpoint of s (a valid end-vertical-rectangle event); (2) The wavefront has already covered portion of s, where s is horizontal (a start-vertical-rectangle event or an invalid V-event); and (3) The wavefront has already covered portion of s, where s is not horizontal (a horizontal-rectangle event).

In Case (1), the handling of the corresponding event (start-vertical-rectangle or horizontal-rectangle event or valid V-event or end-vertical-rectangle event) is similar to processing an ordinary line-segment event:[16,20] The portion of the wavefront between $a(t)$ and $b(t)$ gets finalized and is substituted by the wavecurve of P. For a horizontal-rectangle event, there is one new action to take to handle crossings: for any vertical rectangle Q inducing a crossing left V-vertex on the finalized portion of the wavefront, generate a new V-event (if any) as described in Sec. 4.4.

In Case (2) (start-vertical rectangle event or invalid V-event), a new V-event needs to be generated to predict the next right V-vertex along s (if any). Among the rectangles forming the wavefront, only the rectangle Q with the rightmost west edge (largest *w-max* value) among all waves between $a(t)$ and $b(t)$ may induce a right V-vertex on s (see Lemma 3). Thus, Q can be efficiently identified by binary search on the augmented wavefront using the *w-max* augmentation value of the waves between $a(t)$ and $b(t)$. If a V-event is produced by Q, it could be invalid because the first right V-vertex along s may be induced by a vertical rectangle R that has not yet been inserted in the wavefront. This is possible even in the non-crossing case, where R has already been considered, but R may not have been inserted to the wavefront yet due to the existence of an unprocessed V-event for R.

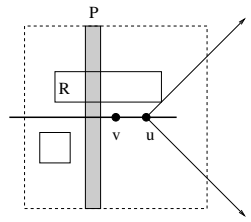

Fig. 8. Proof of Lemma 3.

To avoid generating unnecessary invalid V-events, we use the dominating sequence information form Phase 1 and generate an appropriate V-event choosing between Q and $ds(P)$ as described in Sec. 4.4. The following lemma ensures that Q is correctly identified among the rectangles in the wavefront, and that no V-vertex can exist on s between v and the potential V-vertex induced by Q.

Lemma 3. *Let P be a vertical rectangle and R be the rectangle in $square(P, v)$ with the rightmost west edge among all rectangles enclosed in $square(P, v)$ (see Fig. 8). At time $t = priority(v)$, R^w induces the wave between $a(t)$ and $b(t)$ of maximum w-max (i.e., $Q = R$). Let u be the right V-vertex of $b_h(P, R)$, if $R^w < P^w$, and u be the right endpoint of s otherwise. Then $reg(P) \cap \overline{vu} = \emptyset$, i.e., no Voronoi V-vertex can occur on the portion of the core segment $\overline{vu}$.*

Proof. If $R^w > P^w$, then R must be entirely enclosed in $square(P, x)$ for any point x of s to the right of v. Otherwise, let u be the center of $square(P, R^w)$. Then, for any point x on $\overline{vu}$, R must be entirely enclosed in $square(P, x)$. Thus, in both cases, $reg(P) \cap \overline{vu} = \emptyset$. It remains to show that $Q = R$, where Q is the rectangle determined by the algorithm.

Assuming that $square(P, v)$ contains at least one rectangle other than P, the wavefront must have already covered v at time t. Thus, $r_1(t)$ and $r_2(t)$ intersect the wavefront as shown in Figs. 7(b) and 7(c). Note that $r_1(t)$ and $r_2(t)$ correspond to the diagonals of $square(P, v)$. Since the wavefront is monotone with respect to any line of slope ± 1, the portion of the wavefront between $a(t)$ and $b(t)$ must be entirely included within the triangle $\Delta(t)$ formed by $r_1(t), r_2(t)$, their intersection point v, and l_t. We first argue that R must have a wave within Δ. This is because the $+1$-slope (resp., -1-slope) ray of wavecurve of R must be below $r_1(t)$ (resp. above $r_2(t)$). In addition, since R is entirely enclosed in $square(P, v)$, the vertical portion of its wavecurve (if it exists) or the apex of its V-curve (if the vertical portion does not yet exist) must be to the right of the vertical line through v.

We now argue that only rectangles enclosed within $square(P, v)$ may have a wave within Δ. Let T be a rectangle in the wavefront that is not enclosed in $square(P, v)$ ($priority(T) < t$). If $T^n > P^n$, then the $+1$-slope ray of the wavecurve of T must be above $r_1(t)$. Thus, the entire wavecurve of T must be outside Δ in this case. Similarly, if $T^s < P^s$, then the -1-slope ray of the wavecurve of T must be below

$r_2(t)$, i.e., the entire wavecurve of T must be outside Δ. If $T^w < square(P, v)^w$, then the vertical (thus, the entire) portion of the wavecurve of T must be to the left of the vertical line through v. In all cases, no portion of the wavecurve of T can be within Δ at time t.

By the above arguments, the wave of R must have the maximum *w-max* value among all waves in Δ, which corresponds to Q. Thus, $R = Q$. □

A horizontal-rectangle event for a horizontal rectangle P (or a square P) is processed at time $t = priority(P)$. At this event we need to identify the intersections (V-vertices) of the wavefront with the vertical core segment s (if any). At time t, the wavefront may intersect s a number of times, where each intersection corresponds to a V-vertex, among which only the first and last one may be non-crossing. If there are no intersections because the wavefront has not reached any portion of s yet, then we have Case (1) at $t = priority(P)$; otherwise, we have Case (3). Any portion of the wavefront to the left of s is finalized and gets substituted by the wavecurve of P as fragmented by the V-vertices and their incident spike bisectors. For any left crossing V-vertices on the finalized portion of the wavefront, V-events need to get generated.

To identify the V-vertices on s efficiently, we use the *x-min* value of the augmentation. Let r be the breakpoint of minimum *x-min* value between $a(t)$ and $b(t)$. If r is to the right of s, then s must be entirely covered by the wavefront and $reg(P) = \emptyset$. Otherwise, trace the wavefront sequentially, starting at r, until the first intersections above and below r are determined. The intersection above (resp., below) corresponds to a *down* (resp., *up*) V-vertex v (resp., u). Repeat the process for the portions of the wavefront above v and below u until all intersections are determined. Any time the *x-min* value of a portion of the wavefront is to the right of s, this portion can be eliminated as it contains no intersections with s. Correctness in the handling of a horizontal-rectangle event follows easily.

At each event, we also need to update the augmentation values of the wavefront.

4.4. *Generating and bounding V-events*

Let P be a vertical rectangle of core segment s, and let Q be a candidate rectangle in $square(P, v)$, where v is a point on s. Assuming that $ds(P)$ is available from Phase 1, we generate the next V-event on s, choosing between Q and $ds(P)$, while updating $ds(P)$, as follows:

Algorithm-Generate-V-Event(P, $ds(P)$, Q):
begin

If $ds(P) = \emptyset$, generate a V-event for (P, Q) and return.
Let R be the last rectangle in $ds(P)$;
If R is non-crossing with P then

If $Q^w < R^w$, generate a V-event for (P, R);
Else, generate a V-event for (P, Q);
Delete R from $ds(P)$ and return.

While $ds(P) \neq \emptyset$ do (all rectangles in $ds(P)$ must be crossing with P)
Let R be the last rectangle in $ds(P)$;
If $Q^w < square(P, R^e)^w$, generate a V-event for (P, Q) and return.
Else if $Q^w \leq R^w$, generate a V-event for (P, R); delete R from $ds(P)$ and return;
Else delete R from $ds(P)$; if $ds(P) = \emptyset$, generate a V-event for (P, Q);

end

Lemma 4. *Using the dominating sequences of vertical rectangles computed in Phase* 1*, the total number of V-events that may be produced throughout the algorithm is* $O(n + m')$.

Proof. Let P be a vertical rectangle of core segment s. By Lemma 2, the only vertical rectangles that may generate V-events on s are the rectangles of $ds(P)$. Thus, any other such rectangle must be horizontal (or square). Let $Q_1, \ldots, Q_k$ be a sequence of non-vertical rectangles that induce right V-events on s ordered as $Q_i^w < Q_{i+1}^w$, $i \geq 1$. Among them, only Q_1 can be non-crossing with P, because Q_1 can be present in the wavefront when P is considered at time $t = priority(P)$. Since we always select the rectangle with the rightmost west edge, among all the rectangles in the wavefront dominated by P, rectangle Q_{i+1} is enclosed in $square(P, Q_i^w)$, while rectangle Q_{i+2} is not, for any $i \geq 1$. (Otherwsise we would select Q_{i+2} instead of Q_{i+1}). Thus, the list of odd indexed rectangles $(Q_1, Q_3, \ldots)$, except possibly Q_1, and the list of even indexed rectangles $(Q_2, \ldots)$, must each form a staircase for P. The dominating sequence of P is also a staircase. Each sequence cannot exceed the number of potentially essential crossings for P, and hence, the same holds for the maximum number of invalid V-events generated by each on s. Since there are at most three such sequences for P, the result follows. □

Using a standard analysis for plane sweep and the augmentation of the wavefront, which can handle the horizontal-rectangle event and the generation of V-events efficiently, the time complexity of Phase 2 is $O((n + m + E) \log n)$, where E is the number of invalid V-events. By Lemma 4 and Phase 1, we conclude.

Theorem 2. *HVD*(S) *can be computed by plane sweep in* $O((n + m') \log n)$ *time and* $O(n + m')$ *space. In the case of rectangles with a small number of crossings (e.g.,* $O(n)$*), this is an optimal* $O(n \log n)$*-time and* $O(n)$*-space algorithm.*

5. Extending to Clusters of Axis Parallel Line Segments

The algorithms to compute the L_∞ Hausdorff Voronoi diagram of point clusters directly extends to a set S of clusters of axis-parallel line-segments. The distance

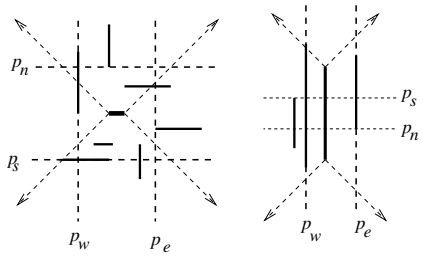

Fig. 9. The farthest Voronoi diagram of axis parallel segments.

between a point t and a line-segment s is $d(t,s) = \min\{d(t,x), \forall x \in s\}$. For a cluster P, its Hausdorff Voronoi region is defined as usual,

$$hreg(P) = \{x \mid d_f(x,P) < d_f(x,Q), \forall Q \in S\},$$

where $d_f(t,P)$ is the farthest distance between t and all line-segments in P, $d_f(t,P) = \max\{d(t,s), \forall \text{ line-segment s}, s \in P\}$.

The L_∞ farthest Voronoi diagram of P, $FVD(P)$, remains similar to the one of points consisting of exactly four unbounded regions, each one induced by an axis-parallel line $p_i, i = n, e, s, w$, called a *boundary line* (see Fig. 9). In particular, the north (resp., south) region of $FVD(P)$ is induced by the horizontal line p_s (resp., p_n) through the bottommost (resp. topmost) upper (resp., lower) endpoint of all line segments in P, and the east (resp., west) region is induced by a vertical line p_w (resp. p_e) through the leftmost (resp., rightmost) right (resp., left) endpoint of all line segments in P. In Fig. 9, the four bounding lines are illustrated as axis-parallel dashed lines. For any point t in the north (resp., south) region, $d_f(t,P) = d(t,p_s)$ (resp., $d_f(t,P) = d(t,p_n)$); equivalently for the east and west regions. The set of boundary lines $\{p_n, p_e, p_s, p_w\}$ is called the *boundary set* and it fully characterizes $FVD(P)$, similarly to the way the minimum enclosing rectangle of a point cluster R characterizes $FVD(R)$.[19] Once the boundary set of P is available, individual cluster segments have no further influence.

Let p_i denote both the boundary line and its main coordinate, $i = n, e, s, w$. Typically, $p_n \geq p_s$ and $p_e \geq p_w$, in which case the boundary set of P is equivalent to the rectangle formed by the common intersection of the four bounding lines (see Fig. 9(a)). It is possible, however, that $p_n < p_s$ (resp., $p_e < p_w$) in case of vertical (resp., horizontal) line segments, in which case the corresponding rectangle is not relevant (see Fig. 9(b)). Nevertheless, each boundary line $p_i, i = n, e, s, w$, can be treated equivalently to the rectangular edge P^i of the previous sections; thus, the algorithms can directly apply using the boundary sets instead of enclosing rectangles.

For completeness, we adapt definitions regarding rectangles to include boundary sets in order to naturally extend the algorithms to clusters of axis-parallel line segments. A cluster P of axis-parallel line segments is considered vertical if its core

segment is horizontal; otherwise, it is considered horizontal. It is easy to see that P is vertical (resp., horizontal) if and only if $p_n - p_s > p_e - p_w$ (resp., $p_n - p_s \leq p_e - p_w$), similarly to a vertical (resp., horizontal) rectangle. The *length* of P is $\max\{p_n - p_s, p_e - p_w\}$. The *square*$(P,v)$ centered on a point v of the core segment of P, having side equal to the length of P, is regarded as a *minimum enclosing square* of the boundary set of P. For a vertical (resp., horizontal) cluster *square*(P,v) is delimited by p_n, p_s (resp., p_e, p_w) as in the case of a normal rectangle. If *square*(P,v) is also delimited by the boundary line q_i of a cluster Q, then it is denoted as *square*(P, q_i) and its center corresponds to a V-vertex of the Hausdorff bisector between P and Q. Assuming (without loss of generality) that $p_n > q_n$, two clusters P, Q are called crossing if $p_s < q_s$, $q_w < p_w$ and $p_e < q_e$.

The Hausdorff bisector $b(P, Q)$ is a subgraph of $FVD(P \cup Q)$. In particular, $b(P, Q)$ consists of the edges in $FVD(P \cup Q)$ corresponding to bisectors of a boundary line of P and a boundary line of Q. In case of collinear bounding lines, a choice can be made based on the conventions used to break ties, similarly to Fig. 2. The Hausdorff bisector of two crossing clusters consists of two disconnected portions each one apexed at a V-vertex. For a vertical cluster P, we have $p_n > p_s$, thus, the definition of the dominating sequence of a vertical cluster remains the same as in the case of a vertical rectangle, where every P^i in Def. 3 corresponds to the bounding line p_i for any $i = n, e, s, w$.

Using the above definition adaptations, both plane sweep phases of Secs. 3 and 4 can be directly applied to the boundary sets, and thus, the same algorithms can be used to compute the Hausdorff Voronoi diagram of clusters of axis parallel line-segments.

6. Conclusion

We revisited the Hausdorff Voronoi diagram in the L_∞ metric and the plane-sweep approach for its construction. We gave a simple two-pass plane sweep, which runs in $O((n + m') \log n)$ time and $O(n + m')$ space, where n is the number of clusters and m' is a number of specially crossing pairs of clusters. The algorithm is optimal ($O(n \log n)$ time and $O(n)$ space) for families of clusters with a small ($O(n)$) number of crossings. The L_∞ Hausdorff Voronoi diagram and its variants have offered the basis for an industrial VLSI CAD tool that computes the probability of fault (POF) for open faults and via-blocks in VLSI designs due to random defects occurring during the manufacturing process.[26] For information on the Voronoi-based approach to VLSI critical area extraction, see e.g., Ref. 18 and references therein.

References

1. M. Abellanas, G. Hernandez, R. Klein, V. Neumann-Lara and J. Urrutia, A combinatorial property of convex sets, *Discr. Comput. Geom.* **17** (1997) 307–318.
2. M. Abellanas, F. Hurtado, C. Icking, R. Klein, E. Langetepe, L. Ma, B. Palop and V. Sacristán, The farthest color Voronoi diagram and related problems, Technical

report, Rheinische Friedrich Wilhelms Universität Bonn (2006), *Abstracts* 17*th European Workshop Comput. Geom.* (2001), pp. 113–116.
3. F. Aurenhammer, R. Klein and D. T. Lee, *Voronoi Diagrams and Delaunay Triangulations* (World Scientific Publishing Company, Singapore, 2013).
4. P. Cheilaris, E. Khramtcova, S. Langerman and E. Papadopoulou, A randomized incremental approach for the Hausdorff Voronoi diagram of non-crossing clusters, *Proc.* 11*th Latin American Theoretical Informatics Symposium* (LATIN 2014) (2014), pp. 96–107.
5. O. Cheong, H. Everett, M. Glisse, J. Gudmundsson, S. Hornus, S. Lazard, M. Lee and H.-S. Na, Farthest-polygon Voronoi diagrams, *Comput. Geom.* **44**(4) (2011) 234–247.
6. F. Dehne and R. Klein, The big sweep: On the power of the wavefront approach to Voronoi diagrams, *Algorithmica* **17** (1997) 19–32.
7. F. Dehne, A. Maheshwari and R. Taylor, A coarse grained parallel algorithm for Hausdorff Voronoi diagrams, *Proc.* 2006 *Int. Conf. Parallel Processing* (2006), pp. 497–504.
8. H. Edelsbrunner, L. Guibas and M. Sharir, The upper envelope of piecewise linear functions: Algorithms and applications, *Discr. Comput. Geom.* **4** (1989) 311–336.
9. D. P. Huttenlocher, K. Kedem and M. Sharir, The upper envelope of Voronoi surfaces and its applications, *Discr. Comput. Geom.* **9** (1993) 267–291.
10. S. J. Fortune, A sweepline algorithm for Voronoi diagrams, *Algorithmica* **2** (1987) 153–174.
11. R. Klein, *Concrete and Abstract Voronoi Diagrams*, Lecture Notes in Computer Science, Vol. 400 (Springer-Verlag, 1989).
12. R. Klein, E. Langetepe and Z. Nilforoushan, Abstract Voronoi diagrams revisited, *Comput. Geom. Theor. Appl.* **42**(9) (2009) 885–902.
13. R. Klein, K. Mehlhorn and S. Meiser, Randomized incremental construction of abstract Voronoi diagrams, *Comput. Geom. Theor. Appl.* **3** (1993) 157–184.
14. E. M. McCreight, Priority search trees, *SIAM J. Comput.* **14** (1985) 257–276.
15. A. Okabe, B. Boots, K. Sugihara and S. N. Chiu, *Spatial Tessellations: Concepts and Applications of Voronoi Diagrams*, 2nd edn. (John Wiley, 2000).
16. E. Papadopoulou, Critical area computation for missing material defects in VLSI circuits, *IEEE Trans. Comp.-Aided Design* **20**(5) (2001) 583–597.
17. E. Papadopoulou, The Hausdorff Voronoi diagram of point clusters in the plane, *Algorithmica* **40** (2004) 63–82.
18. E. Papadopoulou, Net-aware critical area extraction for opens in VLSI circuits via higher-order Voronoi diagrams, *IEEE Trans. Comp.-Aided Design* **30**(5) (2011) 704–716.
19. E. Papadopoulou and S. K. Dey, On the farthest line segment Voronoi diagram, *Int. J. Comput. Geom. Appl.* **23**(6) (2013) 443–460.
20. E. Papadopoulou and D. T. Lee, The L_∞ Voronoi diagram of segments and VLSI applications, *Int. J. Comput. Geom. Appl.* **11**(5) (2001) 503–528.
21. E. Papadopoulou and D. T. Lee, The Hausdorff Voronoi diagram of polygonal objects: A divide and conquer approach, *Int. J. Comput. Geom. Appl.* **14**(6) (2004) 421–452.
22. E. Papadopoulou and J. Xu, The L_∞ Hausdorff Voronoi diagram revisited, *Proc.* 8*th Int. Symp. Voronoi Diagrams in Science and Engineering* (ISVD 2011), IEEE-CS (2011), pp. 67–74.
23. F. P. Preparata and M. I. Shamos, *Computational Geometry: An Introduction* (Springer-Verlag, New York, 1985).
24. Xuehou Tan, Optimal computation of the Voronoi diagram of disjoint clusters, *Inf. Process. Lett.* **79**(3) (2001) 115–119.

25. J. Xu, L. Xu and E. Papadopoulou, Computing the map of geometric minimal cuts, *Algorithmica* **68** (2014) 805–834.
26. Voronoi CAA: Voronoi Critical Area Analysis. IBM VLSI CAD Tool, Department of Electronic Design Automation, IBM Microelectronics Division, Burlington, VT. Distributed by Cadence. Patents: US6178539, US6317859, US7240306, US7240306, US7752589, US7752580.

Appendix

Compute dominating sequences of vertical rectangles
begin

- Sort vertical rectangles in decreasing order of west edge into a list L.
- For $P \in L$, create $p = point(P) = (p_x, p_y, p_z)$, where $p_x = P^n$, $p_y = -P^s$, $p_z = P^w$; let $R(p) = P$; initialize $ds(P) = \emptyset$. Let the expiration time of P be $square(P, P^e)^w$.
- Initialize a dynamic priority search tree T to $\emptyset$.
- Initialize a max-heap H to $\emptyset$. H keeps pairs of crossing rectangles (P, R), such that the expiration time of P has not been reached yet and R is the last rectangle in $ds(P)$ discovered so far. Pair (P, R) corresponds to $Q = square(P, R^e)$. H is organized according to the maximum west edge of $square(P, R^e)$.
- Process rectangles in $L \cup H$ in decreasing order of west edge. Process also the expiration times of original rectangles in decreasing order. For each rectangle R in $L \cup H$ do
  - If $R \in L$, query T for range $J(r)$, $r = point(R)$.
    For each point p in $J(r)$ do
    - Insert R in $ds(P)$, where $P = R(p)$.
    - If R is crossing P, consider $Q = square(P, R^e)$, and insert Q in H. Let $point(Q) = (p_x, p_y, p'_z)$, where $p'_z = Q^w$. Let $R((p_x, p_y, p'_z)) = P$, and set the expiration time of $point(Q)$ to the one of P.
    - Delete p from T.
  - Insert $point(R)$ into T ($R \in L \cup H$).
  - If the expiration time of a rectangle is reached, delete the corresponding point from T.

end