A Sweepline Algorithm for Higher Order Voronoi Diagrams*

Maksym Zavershynskyi  
Faculty of Informatics, Università della Svizzera italiana  
Lugano, Switzerland  
Email: maksym.zavershynskyi@usi.ch

Evanthia Papadopoulou  
Faculty of Informatics, Università della Svizzera italiana  
Lugano, Switzerland  
Email: evanthia.papadopoulou@usi.ch

Abstract—We present an algorithm to construct order-$k$ Voronoi diagrams with a sweepline technique. The sites can be points or line segments. The algorithm has $O(k^2n \log n)$ time complexity and $O(nk)$ space complexity.

I. INTRODUCTION

Given a set of $n$ simple geometric objects in the plane, called sites, the order-$k$ Voronoi diagram of $S$, $V_k(S)$ is a partitioning of the plane into regions, such that every point within a fixed order-$k$ Voronoi region has the same set of $k$ nearest sites. For $k = 1$ this is the nearest-neighbors Voronoi diagram and for $k = n-1$ the farthest-site Voronoi diagram.

The structural complexity of the order-$k$ Voronoi diagram in the $L_p$ metric for both points and disjoint line segments is $O(k(n-k))$ [11], [15]. For line segments, unlike points, order-$k$ Voronoi regions may be disconnected, in particular one Voronoi region may get disconnected into $\Omega(n)$ disjoint faces [15]. Despite disconnected regions the structural complexity remains asymptotically the same. For points or axis-parallel line segments in the $L_\infty$ metric the structural complexity is $O\left(n \min\{k(n-k), (n-k)^2\}\right)$ [13].

A standard simple technique to compute the order-$k$ Voronoi diagram is an iterative construction, where the order-$k$ Voronoi diagram is computed from the order-$(k-1)$ diagram, for increasing values of $k$, in $O(k^2n \log n)$ time and $O(k^2(n-k))$ space [11]. For points in Euclidean plane, more sophisticated techniques have been developed based on randomized approaches and duality between the order-$k$ Voronoi diagram and the $k$-level of planes in $R^3$, see e.g. [1], [4], [5], [16]. These algorithms exploit the fact that the sites are points, they construct the $k$-level in an arrangement of planes, and they are not simple to generalize to line segments, to the best of our knowledge. The best expected running time is $O(n \log n + nk2^{c \log^2 k})$ [16], for a constant $c$, which is near-optimal, however, mostly of theoretical interest as noted in [16].

In this paper we investigate the plane sweep paradigm for the construction of higher order Voronoi diagrams of polygonal sites (including points and line segments) as a simple alternative to the iterative construction. Plane sweep has not been considered so far for the construction of higher order Voronoi diagrams. It is essentially an iterative construction, which, does not require the pre-computation and storage of lower order Voronoi diagrams, but achieves the construction in a single plane sweep pass of the input data. This ability can be useful in practice, especially when information involving all order-$i$, $i \leq k$, Voronoi diagrams is required. For example, in [14], the geometric min-cut problem in a VLSI layout is addressed by iteratively computing higher order Voronoi diagrams of (weighted) line segments that represent polygons.

Information of low-order Voronoi diagrams is important in this application and the importance grows weaker fast as $k$ increases. However, an order-1 diagram is not sufficient. A direct plane sweep construction of the order-$i$ Voronoi diagram, $i \leq k$, for small $k$, is valuable to this application (see [14]).

Our algorithm is based on plane sweep [8], which constructs the nearest neighbor Voronoi diagram in $O(n \log n)$ time and $O(n)$ space. It constructs the order-$i$ Voronoi diagrams, $i \leq k$, in $O(k^2n \log n)$ time and $O(nk)$ space.

II. PRELIMINARIES

In this abstract we first describe the algorithm for disjoint line segments (including points). In Section V we extend to line segments forming a planar straight-line graph. For the definition of the order-$k$ Voronoi diagram in this case see [15].

Let $S = \{s_1, s_2, \ldots, s_n\}$ be a set of $n$ disjoint line segments in $R^2$, called sites. For disjoint line segments we make a general position assumption that no more than three sites can touch the same circle. The Euclidean distance between two points $p, q$ is denoted as $d(p, q)$. The distance between point $p$ and line segment $s$ is the minimum Euclidean distance $d(p, s) = \min_{q \in s} d(p, q)$. The order-$k$ Voronoi diagram $V_k(S)$ of $S$ is a partitioning of the plane into order-$k$ Voronoi regions such that every point within a fixed region $V_k(H, S)$ has the same $k$ nearest sites $H$.

$$V_k(H, S) = \{x | \forall s \in H \forall t \in S \setminus H \ d(x, s) < d(x, t)\}$$ (1)

A maximal interior-connected subset of a region is called a face.

Let $l$ be a horizontal line such that the halfplane $l^+$ above $l$ intersects at least $k$ line segments in $S' \subseteq S$. The locus of points equidistant from line segment $s \in S'$ and $l$ is denoted as $w(s)$ and it is called the wave-curve of $s$ (see Fig. 1). Consider an arrangement $A$ of wave-curves $w(s)$, $s \in S'$. The $i$-level of $A$ is the set of points $x$, such that $x$ belongs to some wave-curve $w(s)$ and has $i$ wave-curves below, including $w(s)$ [7]. The intersection point of two wave-curves $w(s_1)$ and $w(s_2)$ is denoted $w(s_1, s_2)$.

---

* Supported in part by the Swiss National Science Foundation (SNF) grant 200021-127137 and the ESF EUROCORES program EuroGIGA-VORONOI, SNF project 20GG21-134355.
on level \( i \) has exactly \( i \) or \( i + 1 \) wave-curves below, including \( w(s_1) \) and \( w(s_2) \); \( x \) is called a breakpoint of level \( i \). The \( i \)-level of \( A \) is denoted as \( A_i \). Level \( A_i \) is a sequence of waves, each wave being a portion of a wave-curve between two consecutive intersection points on level \( i \). A single wave-curve may contribute more than one wave to \( A_i \). The ordinary wavefront in the standard plane sweep construction of Voronoi diagrams is the 1-level of \( A \).

The idea of the algorithm is to sweep the plane with the horizontal line while maintaining levels \( A_1, \ldots, A_k \). First we prove that the portion of the order-\( k \) Voronoi diagram above level \( A_k \) does not change as the sweepline moves down (Lemma 1) and the breakpoints of level \( A_k \) move along the edges of the order-\( k \) Voronoi diagram (Corollary 1). Then we describe the discrete event points that change the topological structure of levels \( A_1, \ldots, A_k \) (Lemmata 3 and 4). Finally we show that in order to maintain level \( A_k \) it is sufficient to track topological changes in levels \( A_1, \ldots, A_k \).

**Lemma 1:** Let \( l \) be a horizontal line and let \( S' \subseteq S \) be a set of sites that intersect \( l^+ \). Consider a point \( x \) above \( A_i \) and let \( x \in V_i(H, S') \) then \( x \in V_i(H, S) \).

**Proof:** If we consider the disk centered at \( x \) that touches line \( l \) the claim of the lemma will be easy to see.

In other words the part of \( V_i(S) \) above \( A_i \) is not affected by the sites strictly below \( l \).

**Lemma 2:** Let \( l \) be a horizontal line and let \( S' \subseteq S \) be a set of sites that intersect \( l^+ \). Consider a point \( x, x \in A_i \) and \( x \) belongs to a single wave-curve \( w(s_i) \). Let \( w(s_1), \ldots, w(s_{i-1}) \) be a set of wave-curves that are below \( x \). Then \( x \in V_i(H, S) \).

**Proof:** Let \( w(s') \) be a wave above \( x \) then \( d(x, s') > d(x, l) \) and \( d(x, l) > d(x, s_j) \), for \( j = 1, \ldots, i \). This and Lemma 1 imply that any site in \( H \) is closer to \( x \) than any site in \( S \setminus H \). Therefore \( x \in V_i(H, S) \).

The bisector of two segments \( s_j \) and \( s_h \) is the locus of points equidistant from both, \( b(s_j, s_h) = \{ x \mid d(x, s_j) = d(x, s_h) \} \). The edge of the order-\( i \) Voronoi diagram that bounds two adjacent faces \( F_j \) and \( F_h \) of regions \( V_i(H \cup \{s_j\}, S) \) and \( V_i(H \cup \{s_h\}, S) \) is portion of bisector \( b(s_j, s_h) \). The following corollary implies that while the sweepline moves down, the intersections of two levels \( A_i \) and \( A_i+1 \) move along the Voronoi edges of the order-\( i \) Voronoi diagram.

**Corollary 1:** Let \( x \) be an intersection point of two levels \( A_i \) and \( A_i+1 \) incident to wave-curves \( w(s_j) \) and \( w(s_h) \). Let \( H \) be a set of sites that correspond to the wave-curves below \( x \). Then \( x \) belongs to the Voronoi edge bounding two faces of regions \( V_i(H \cup \{s_j\}, S) \) and \( V_i(H \cup \{s_h\}, S) \).

Voronoi vertices in \( V_i(S) \) are classified into new and old [11], [15]. A Voronoi vertex of \( V_i(S) \) is called new (respectively old) if it is the center of a disk that touches 3 sites and its interior intersects exactly \( i - 1 \) (respectively \( i - 2 \)) sites. By the definition of the order-\( i \) Voronoi diagram every Voronoi vertex of \( V_i(S) \) is either new or old. A new Voronoi vertex in \( V_i(S) \) is an old Voronoi vertex in \( V_{i+1}(S) \). Under the general position assumption, an old Voronoi vertex in \( V_i(S) \) is a new Voronoi vertex in \( V_{i-1}(S) \).

During the sweeping process events occur when three wave-curves intersect at a common point. The following lemma ties up such an event with a Voronoi vertex of the order-\( i \) Voronoi diagram of \( S \).

**Lemma 3:** Let \( l, H \) and \( A \) be a horizontal line, a set of line segments that intersect \( l^+ \) and an arrangement of wave-curves created by the line segments in \( H \). Let \( x \) be an intersection point of three waves (see Fig. 1) such that \( x \) belongs to three levels \( A_{i+2}, A_{i+1} \) and \( A_i \). Then \( x \) is a new Voronoi vertex of \( V_i(S) \) and an old Voronoi vertex of \( V_{i+1}(S) \).

**Proof:** Let \( x \) be an intersection point of \( w(s_1), w(s_2) \) and \( w(s_3) \) then \( d(x, s_1) = d(x, s_2) = d(x, s_3) = d(x, l) \). Lemma 2 implies the claim.

Since wave-curves are \( x \)-monotone, level \( A_i \) is also an \( x \)-monotone curve. Corollary 1 implies that the breakpoints of \( A_i \) lie on Voronoi edges of the final diagrams \( V_i(S) \) and \( V_{i-1}(S) \). As line \( l \) moves down the waves forming \( A_i \) change their shapes continuously. We maintain the sweepline status for level \( A_i \) and we are interested in simulating the discrete event points that change the topological structure of \( A_i \) and Voronoi diagrams \( V_i(S) \) and \( V_{i-1}(S) \). There are two types of events:

**Site-event:** Line \( l \) hits a new site \( s \); then a new wave-curve \( w(s) \) should be added to \( A \). \( w(s) \) is added to all levels of \( A \), in level-\( i \). Adding \( w(s) \) is similar to inserting parabolas in the nearest neighbor Voronoi diagram case [8].

**Circle-event:** Three wave-curves intersect at a common point. A circle event corresponds to a new vertex at some level-\( i \) and an old vertex at level-\( i + 1 \) (Lemma 3).

**Lemma 4:** Consider a circle-event at point \( x \in A_i \) such that \( x \) is a new Voronoi vertex of \( V_i(S) \). Then at this event a single wave disappears from \( A_i \).
The circle-event triggers topological changes on three levels $A_i, A_{i+1}, A_{i+2}$\(^1\). Thus to handle a circle-event we simulate the changes on all three levels. At the instant when the three wave-curves intersect at a common point the sweepline touches the bottom-most point of the disk that is tangent to three sites. For every triple of consecutive waves we create a circle-event in advance and we process it when the sweepline touches the bottom-most point of the disk. Let $w(s_1), w(s_2), w(s_3)$ be three consecutive waves that generate such event on level $A_i$ (see Fig. 2 (a)). We delete the middle wave $w(s_2)$ from the level $A_i$. On level $A_{i+1}$ the waves appear as $w(s_2), w(s_1), w(s_3), w(s_2)$, so we switch the order of waves $w(s_1)$ and $w(s_3)$. On level $A_{i+2}$ the waves appear as $w(s_3), w(s_1), w(s_3)$ so we add the wave $w(s_2)$ in between.

A new wave may be introduced to a level $A_i$ by a site-event or from the levels below $A_i$. In order to maintain $A_i$ during the sweepline it is sufficient to handle all site-events and all circle-events that appear on levels $1, \ldots, i$.

### III. The Algorithm

We sweep the plane with the horizontal line $l$ while maintaining levels $A_1, \ldots, A_k$. Each level $A_i$ corresponds to a sequence of waves $w(s_1), w(s_2), \ldots$. For each level $A_i$ we store the ordered list $L_i$ of corresponding sites allowing search/insertion/deletion in logarithmic time. We also store a priority queue $Q$ of events ordered lexicographically by $y$-coordinate. A site-event occurs when line $l$ hits a new line segment. We order site-events by their $y$-coordinate of the topmost endpoint. We also create site-events for all the bottommost endpoints.

\(^1\)This is under the general position assumption. If we remove the general position assumption then the changes may occur to more than 3 levels.

We store circle-event as a disk that touches three line segments and we order them by their $y$-coordinate of the bottom-most point of the disk. We create circle-events every time the new consecutive triple of waves appears at some level after the site-events and circle-events. Every time the adjacency relations of levels $A_1, \ldots, A_k$ change we need to create new events for new consecutive triples of waves and remove those events that do not correspond to consecutive triples anymore. For brevity we denote as $\text{update}_\text{triplets}(L_i, r_j, r_h)$ an operation that updates those triples that involve positions $r_1, \ldots, r_h$ in list $L_i$. Operation $\text{substitute}(L_i, r, [a, b, c, [d, e, f]])$ substitutes in list $L_i$ at position $r$ subsequence $a, b, c$ with subsequence $d, e, f$. We output order-$i$ Voronoi diagram as a set of Voronoi vertices and incident bisectors.

1: Initialize $Q = [s_1, \ldots, s_n]$ sorted by $y$-coordinate of their topmost endpoint.
2: Extract the first site $s_1$.
3: Initialize $L_1 = [s_1]$ and $L_i = []$ for $i = 2, \ldots, k$.
4: while $Q$ non empty do
5: $x \leftarrow \text{top most}(Q)$
6: if $x$ is a new site then
7: Search for the wave in $L_1$ to which $x$ belongs.
8: Let $s \in L_1$ be the corresponding line segment and $r$ be the position in list $L_1$.
9: substitute($L_1, r, [s, x, s])$
10: update_\text{triplets}($L_1, r, r+2$)
11: $s' \leftarrow s$.
12: for $i = 2, \ldots, k$ do
13: if $L_i$ is empty then
14: $L_i = [x, s', x']$
15: exit for loop
16: else
17: Search for a position in $L_i$ to which $x$ belongs.
18: Let $s \in L_i$ be the corresponding line segment and $r$ be the position in list $L_i$.
19: substitute($L_i, r, [s, x, s']$)
20: update_\text{triplets}($L_i, r, r+4$)
21: $s' \leftarrow s$.
22: end if
23: end for
24:
25: Let $x$ be a disk tangent to line segments $s_1, s_2, s_3$ and the event occurs at levels $A_i, A_{i+1}, A_{i+2}$ at positions $r, r', r''$, respectively (if $i+1 > k$ or $i+2 > k$ then the corresponding values are empty).
26: if $i = k$ or $i+1 = k$ then
27: output $x$ and three bisectors incident to it
28: substitute($L_i, r, [s_1, s_2, s_3], [s_1, s_3]$)
29: update_\text{triplets}($L_i, r, r+2$)
30: if $i+1 \leq k$ then
31: substitute($L_{i+1}, r', [s_2, s_1, s_3, s_2], [s_2, s_3, s_1, s_2]$)
32: update_\text{triplets}($L_{i+1}, r', r'+3$)
33: end if
34: if $i+2 \leq k$ then
35: substitute($L_{i+2}, r'', [s_3, s_1, s_3], [s_3, s_2, s_1]$)
36: update_\text{triplets}($L_{i+2}, r'', r''+2$)
37: end if
38: end if
39: end while

The algorithm maintains the $i$-levels of arrangement $A_i$, $i \leq k$, while sweeping the plane with a horizontal line. Since

Fig. 2: Simulating the circle-event on levels $A_i, A_{i+1}, A_{i+2}$. 

- For brevity we denote as $\text{update}_\text{triplets}(L_i, r_j, r_h)$ an operation that updates those triples that involve positions $r_1, \ldots, r_h$ in list $L_i$. Operation $\text{substitute}(L_i, r, [a, b, c, [d, e, f]])$ substitutes in list $L_i$ at position $r$ subsequence $a, b, c$ with subsequence $d, e, f$. We output order-$i$ Voronoi diagram as a set of Voronoi vertices and incident bisectors. 

1: Initialize $Q = [s_1, \ldots, s_n]$ sorted by $y$-coordinate of their topmost endpoint.
2: Extract the first site $s_1$.
3: Initialize $L_1 = [s_1]$ and $L_i = []$ for $i = 2, \ldots, k$.
4: while $Q$ non empty do
5: $x \leftarrow \text{top most}(Q)$
6: if $x$ is a new site then
7: Search for the wave in $L_1$ to which $x$ belongs.
8: Let $s \in L_1$ be the corresponding line segment and $r$ be the position in list $L_1$.
9: substitute($L_1, r, [s, x, s])$
10: update_\text{triplets}($L_1, r, r+2$)
11: $s' \leftarrow s$.
12: for $i = 2, \ldots, k$ do
13: if $L_i$ is empty then
14: $L_i = [x, s', x']$
15: exit for loop
16: else
17: Search for a position in $L_i$ to which $x$ belongs.
18: Let $s \in L_i$ be the corresponding line segment and $r$ be the position in list $L_i$.
19: substitute($L_i, r, [s, x, s'])$
20: update_\text{triplets}($L_i, r, r+4$)
21: $s' \leftarrow s$.
22: end if
23: end for
24:
25: Let $x$ be a disk tangent to line segments $s_1, s_2, s_3$ and the event occurs at levels $A_i, A_{i+1}, A_{i+2}$ at positions $r, r', r''$, respectively (if $i+1 > k$ or $i+2 > k$ then the corresponding values are empty).
26: if $i = k$ or $i+1 = k$ then
27: output $x$ and three bisectors incident to it
28: substitute($L_i, r, [s_1, s_2, s_3], [s_1, s_3]$)
29: update_\text{triplets}($L_i, r, r+2$)
30: if $i+1 \leq k$ then
31: substitute($L_{i+1}, r', [s_2, s_1, s_3, s_2], [s_2, s_3, s_1, s_2]$)
32: update_\text{triplets}($L_{i+1}, r', r'+3$)
33: end if
34: if $i+2 \leq k$ then
35: substitute($L_{i+2}, r'', [s_3, s_1, s_3], [s_3, s_2, s_1]$)
36: update_\text{triplets}($L_{i+2}, r'', r''+2$)
37: end if
38: end if
39: end while

The algorithm maintains the $i$-levels of arrangement $A_i$, $i \leq k$, while sweeping the plane with a horizontal line. Since
breakpoints of the $i$-level move along the edges of $V_i(S)$ and $V_{i-1}(S)$ the correctness of the algorithm follows. The changes in the $i$-levels of arrangement $A$, $i \leq k$ may be caused by inserting new wave-curves or adjacency changes. Note that even though order-$k$ Voronoi regions of line segments may be disconnected [15], the circle-events handle adjacency changes efficiently.

### IV. Complexity Analysis

**Lemma 5:** The maximum size of queue $Q$ and the maximum total size of lists $L_1, \ldots, L_k$ are $O(nk)$.

**Proof:** Lists $L_1, \ldots, L_k$ correspond to levels $A_1, \ldots, A_k$. Since the wave-curves are Jordan curves the following bound holds [5], [17]:

$$g_{\leq k}(n) = O\left(k^2g_1\left(\left\lfloor n/k \right\rfloor \right)\right) \tag{2}$$

where $g_{\leq k}(n)$ is the maximum complexity of levels $A_1, \ldots, A_k$ and $g_1(m)$ is the maximum complexity of the lower envelope of $m$ wave-curves. The lower envelope of waves corresponds to wavefront of the order-$1$ Voronoi diagram [8], therefore $g_1(m) = O(m)$. Thus the maximum complexity of $A_1, \ldots, A_k, g_{\leq k}(n)$ is equal to $O(nk)$.

The number of site-events is $O(n)$. Every circle-event in event queue $Q$ corresponds to a triplet of adjacent waves at some level $A_i, 1 \leq i \leq k$. Therefore the number of circle-events is proportional to the total size of levels $A_1, \ldots, A_k$, which is $O(nk)$. Thus $Q$ is of size $O(nk)$.

**Theorem 1:** The algorithm can be implemented to run in time $O(k^2n \log n)$ and $O(nk)$ space.

**Proof:** The site-events correspond to the insertion of the new wave-curves in levels $A_1, \ldots, A_k$. The number of new waves is bounded by the number of sites, $O(n)$. When a new line segment intersects a halfplane $l^+$ we insert it in lists $L_1, \ldots, L_k$. This requires a binary search on every list and therefore it takes $O(\log |L_i|)$ per list, where $|L_i|$ denotes the size of the list. Since the maximum complexity of $L_i$ is bounded by the structural complexity of the order-$i$ Voronoi diagram, we need $O(\log (i(n-i))) = O(\log n)$ per level $L_i$, or $O(k \log n)$ for all levels. Therefore it takes $O(nk \log n)$ time to process all the site-events.

The circle-events correspond to the Voronoi vertices of the order-$i$ Voronoi diagrams, $i = 1, \ldots, k$. Every such event requires constant time. Since the number of order-$i$ Voronoi vertices is bounded by $O(i(n-i))$ thus it implies that the total number of circle-events is bounded by $\sum_{i=1}^{k} O(i(n-i)) = O(k^2n)$. Every site-event and circle-event requires an update of the triples that involve the line segments that are adjacent to the places where the changes occurred. Insertions and deletions into the event queue $Q$ require $\log |Q|$ time per each inserted/removed circle-event, where $|Q|$ - is the size of the queue. Lemma 5 implies that the size of the queue is $O(nk)$. Therefore it takes $O(\log (nk)) = O(\log n)$ time per event. And the total running time is $O(k^2n \log n)$.

During the execution of the algorithm we store the event queue $Q$, lists $L_1, \ldots, L_k$ and we output the order-$k$ Voronoi diagram $V_k(S)$. Then Lemma 5 implies the total space complexity.

### V. Line segments forming a planar straight-line graph

In this section we consider line segments that may touch at endpoints, such as line segments forming a simple polygon, more generally line segments forming a planar straight-line graph. This is important for applications involving polygonal objects such as in [14].

Abutting line segments do not satisfy the standard general position assumption that no more than 3 sites can touch the same circle. We make weak general position assumption that no more than 3 elementary sites can touch the same circle\(^2\), where each segment consists of three elementary sites: two endpoints and an open line segment.

Under the weak general position assumption we have two sources of degeneracy: 1) bisectors may contain 2-dimensional portions; 2) bisectors intersect non-transversely. For the order-$1$ Voronoi diagram this is typically addressed efficiently by considering the endpoints and open portion of line segments as different entities [9]. For higher order Voronoi diagrams we cannot simply consider elementary sites as distinct when defining an order-$k$ Voronoi region as this will lead to a different type of order-$k$ Voronoi diagram for disjoint segments. For instance, in case of farthest Voronoi diagram the result is the farthest Voronoi diagram of endpoints which is different from the farthest Voronoi diagram of line segments [3]. We want to keep the information on the shared endpoints without altering the structure of the order-$k$ Voronoi diagram of disjoint line segments (see [15]).

We review some definitions given in the journal version of [15]. A disk centered at point $x$ that intersects (or touches) at least $k$ line segments and has minimal radius is called an order-$k$ disk and is denoted as $D_k(x)$. If $D_k(x)$ touches exactly one elementary site $p$ then it is called a proper order-$k$ disk and is denoted as $D^p_k(x)$. Let $S_k(x)$ denote the set of line segments in $S$ that have a non-empty intersection with $D_k(x)$. It is not hard to see that the order-$k$ disk $D_k(x)$ is unique for every point $x$ on the plane, and thus the same holds for $S_k(x)$.

Following [15], we extend the notion of subset of $S$ of cardinality $k$ as follows.

**Definition 1:** A set $H \subseteq S$ is called an order-$k$ subset if

1. $|H| = k$ (type 1)
2. $|H| > k$ and there exists a proper order-$k$ disk $D^p_k(x)$, where $p$ is an endpoint common to at least two segments, that intersects exactly all line segments in $H$ (type 2). Point $p$ is called the representative of $H$. The order-$k$ subset $H$ of representative $p$ is denoted $H_p$.

The order-$k$ Voronoi region of a type 1 order-$k$ subset $H$ is defined in the ordinary way as given in (1). The order-$k$ Voronoi region of a type 2 order-$k$ subset $H_p$ is defined as

$$V_k(H_p, S) = \{ x \mid \exists D^p_k(x) \wedge S_k(x) = H_p \} \tag{3}$$

\(^2\)In case when line segment $s$ touches disk $D$ at endpoint $p$ and the line passing through $s$ is tangent to disk $D$ it is possible to slightly move disk $D$ to make it intersecting the open partition of $s$. Therefore in this case we consider both endpoint $p$ and the open partition of $s$ to be touching disk $D$. \]
A of wave-curves that intersect non-transversely denote as \( \pi(x) = (\pi^-(x), \pi^0(x), \pi^+(x)) \) sets of line segments that correspond to wave-curves strictly below point \( x \), passing through \( x \) and strictly above \( x \), respectively [7]. We define a \( k \)-level \( A_k \) as the set of points \( x \) such that \( A_k [\pi^-(x)] < k \) and \( |\pi^-(x)\cup\pi^0(x)| \geq k \) (see [7]). \( \pi \) defines an equivalence relation on the plane, where two points \( x \) and \( y \) are equivalent iff \( \pi(x) = \pi(y) \). Equivalence classes partition \( A_k \). The connected components of \( A_k \) that consist of more than a single point are called waves (similarly to the definition of a wave in case of disjoint line segments). Each wave corresponds to equivalence class \( \alpha = (\alpha^-, \alpha^0, \alpha^+) \) where for any point \( x \) on the wave \( \pi(x) = \alpha \). We call a point separating two waves of \( k \)-level a breakpoint. It is easy to see that for every point \( x \) on level \( A_k \) disk \( D_k(x) \) touches line \( l \) and \( S_k(x) = \pi^-(x) \cup \pi^0(x) \). If \( x \) is not a breakpoint then \( D_k(x) \) is a proper order-\( k \) disk. If \( x \) is a breakpoint then \( D_k(x) \) touches \( 2 \) or \( 3 \) elementary sites.

While line \( l \) moves down the breakpoints of level \( A_k \) move along the bisectors of elementary sites. These bisectors correspond to the edges of the order-\( i \) Voronoi diagram, where \( 1 \leq i \leq k \). We have two types of Voronoi edges: external and internal. External Voronoi edges bound order-\( k \) Voronoi regions. Internal Voronoi edges belong to the interior of the order-\( k \) Voronoi regions and they represent the farthest subdivision inside the order-\( k \) Voronoi regions. Type-2 Voronoi regions do not contain internal edges. Let \( x \) be a breakpoint on level \( A_k \) that separates two waves with equivalence classes \( \pi_1 \) and \( \pi_2 \). If \( |\pi_1^0 \cup \pi_1^-| \geq k, \pi_1^2 \cup \pi_2^- > k, \pi_1^0 \cup \pi_1^- = \pi_2^0 \cup \pi_2^- \), \( \pi_1^0 \subseteq I(p) \) and \( \pi_2^0 \subseteq I(p) \) then \( x \) is not on the edge of the order-\( k \) Voronoi diagram, where \( p \) is some endpoint. If \( |\pi_1^0 \cup \pi_1^-| = k, |\pi_1^0 \cup \pi_1^-| = k, \pi_1^0 \cup \pi_1^- = \pi_2^0 \cup \pi_2^- \), then \( x \) is on the internal edge of the order-\( k \) Voronoi diagram. Otherwise \( x \) is on the external edge of the order-\( k \) Voronoi diagram (see Fig. 6).

The algorithm sweeps the plane with the horizontal line while maintaining levels \( A_1, \ldots, A_k \). For each wave on every \( i \)-level we store the set of wave-curves \( \pi^i \) and the number of wave-curves \( b = |\pi^-| \), for \( i = 1, \ldots, k \). A wave may appear on several levels and therefore we store each level \( A_i \) as list \( L_i \) of links to waves and breakpoints to avoid duplication. Each breakpoint is linked to its incident waves and levels. The
Let $p$ be an endpoint that corresponds to a site-event. Let $I^-(p)$ be the set of line segments incident to point $p$ that are below horizontal line $l$. Let $I^+(p)$ be the set of line segments incident to point $p$ that are above horizontal line $l$. Then before line $l$ hits endpoint $p$ the line segments in $I^-(p)$ do not contribute the wave-curves to the arrangement. After line $l$ hits endpoint $p$ the line segments in $I^-(p)$ contribute the wave-curves to the arrangement that intersect transversely between each other. Therefore we can insert the wave-curves of the line segments in $I^-(p)$ one by one using the method described for disjoint line segments.

Consider line segments in $I^+(p)$ and suppose they are ordered from left to right as $s_1, s_2, \ldots, s_m$. Denote the corresponding wave-curves as $w_1, w_2, \ldots, w_m$. Consider the moment before the site-event occurs, and consider the $i$-level of wave-curves above point $p$, where $1 \leq i \leq m$. The $i$-level contains the sequence of waves that have the following $\pi^0$ of their equivalence classes: $w_i, w_{i+1}, w_{i+2}, w_{i+3}, \ldots, w_m, w_{m-i+1}$, see Fig. 7 (a). We handle the site-event by replacing the sequence of waves with the new sequence of waves that have the following $\pi^0$ of their equivalence classes: $w_i, \{w_1, \ldots, w_m\}, \{w_{i-1}, \ldots, w_m\}, \ldots, \{w_1, \ldots, w_m\}, \ldots, \{w_1, \ldots, w_{m-i+1}\}, w_{m-i+1}$, see Fig. 7 (b).

Circle-events have the following two cases: 1) All breakpoints have degree 4, see Fig. 8; 2) One breakpoint has degree 3, see Fig. 9. We handle the circle-event by removing the red waves and adding the new blue waves with equivalence classes depicted on Fig. 8, 9. It is not possible to have more than one breakpoint with degree 3 participating in the same circle-event, since this will correspond to degeneracy with more than 3 elementary sites touching a disk.

We store a priority queue $Q$ of events ordered lexicograph-
Fig. 9: Arrangement of wave-curves (a) before and (b) after the circle-event occurs. One breakpoint has degree 3. The waves are labeled by $b = |\pi^-|$ and $\pi^0$ of the corresponding equivalence classes.

Algorithmically by y-coordinate, where in case of circle-events we take y-coordinate of the bottom-most point of the disk. When we process the event we add the circle-events of new consecutive triples of waves that appear and we remove those circle-events that correspond to the disappeared consecutive triples. Each such event may trigger creation and deletion of a constant number of circle-events per level.

1. Initialize $Q = [p_1, \ldots, p_m]$ with the endpoints sorted by $y$-coordinate.
2. Initialize $L_i = []$ for $i = 1, \ldots, k$.
3. **while** $Q$ non empty **do**
   4. **if** $top(Q)$ is a site-event **then**
      5. $p \leftarrow extract\_top(Q)$
      6. **for** $i = 1, \ldots, |I^+(p)|$ and $L_i$ is not empty **do**
         7. Replace in $L_i$ waves according to Fig. 7
      **end for**
   **end if**
   8. **for all** $s$ line segment in $I^-(p)$ **do**
      9. Add wave-curve $w(s)$ to levels $L_1, \ldots, L_k$
   **end for**
3. **else**
   10. **if** $top(Q)$ is a circle-event **then**
      11. $j \leftarrow 1$
      12. $C[j] \leftarrow extract\_top(Q)$
      13. **while** $C[j].y = top(Q).y$ **do** {Place in $C$ circle-events that occurred simultaneously}
         14. $j \leftarrow j + 1$
         15. $C[j] \leftarrow extract\_top(Q)$
      **end while**
      16. $B \leftarrow$ breakpoints in $C$
      17. $L \leftarrow$ levels passing through breakpoints in $B$
      18. **if** $B$ contains only breakpoints of degree 4 **then**
         19. Replace in levels $L$ waves according to Fig. 8
      **end if**
5. **end if**
8. **end while**

The algorithm maintains $i$-levels of arrangement $A$ of wave-curves that may intersect non-transversely, for $i = 1, \ldots, k$. The breakpoints of $k$-level move along the external and internal edges of the order-$k$ Voronoi diagram of a planar straight-line graph. The algorithm outputs Voronoi vertices and incident Voronoi edges every time it handles events at which Voronoi edges meet.

**REFERENCES**


