Forest-Like Abstract Voronoi Diagrams in Linear Time*

Cecilia Bohler, Rolf Klein, and Chih-Hung Liu
University of Bonn, Institute of Computer Science I, D-53113 Bonn, Germany.

Abstract

Voronoi diagrams are a well-studied data structure of proximity information, and although most cases require \( \Omega(n \log n) \) construction time, it is interesting and useful to develop linear-time algorithms for certain Voronoi diagrams. We develop a linear-time algorithm for abstract Voronoi diagrams in a domain where each site has a unique face and no pair of bisectors intersect outside the domain. Since abstract Voronoi diagrams are a category of Voronoi diagrams, our algorithm works for many concrete Voronoi diagrams.

1 Introduction

Abstract Voronoi diagrams (AVDs, for short) were introduced in Klein [3]. Here, no sites, circles, or distance measures are given. Instead, one takes unbounded curves \( J(p,q) = J(q,p) \) as primary objects, together with the open domains \( D(p,q) \) and \( D(p,q) \) they separate. Abstract Voronoi regions are defined by

\[
\text{VR}(p,S) := \bigcap_{q \in S \setminus \{p\}} D(p,q)
\]

and the abstract Voronoi diagram by

\[
V(S) := \mathbb{R}^2 \setminus \bigcup_{p \in S} \text{VR}(p,S).
\]

The following axioms were required to hold for each subset \( S' \) of \( S \).

(A1) Each curve \( J(p,q) \), where \( p \neq q \), is unbounded. After stereographic projection to the sphere, it can be completed to a closed Jordan curve through the north pole.

(A2) Each nearest Voronoi region \( \text{VR}(p,S') \) is nonempty and pathwise connected.

(A3) Each point of the plane belongs to the closure of a Voronoi region \( \text{VR}(p,S') \).

Certain practical applications only require a specific substructure of the entire diagram or a special kind of Voronoi diagram. Although the construction time is \( \Omega(n \log n) \) for many kinds of Voronoi diagrams, it is still possible to compute a specific part or a special case faster. Aggarwal et al. [1] developed a linear-time algorithm for Euclidean Voronoi diagrams in convex position. Their algorithm further allows to delete a site from Euclidean Voronoi diagrams in time linear to the structural changes, and also speeds up the algorithm for the \( k \)-th-order Voronoi diagram in [6] by a \( O(\log n) \) factor. Later, Klein and Lingas [5] generalized their idea to abstract Voronoi diagrams where a Hamiltonian path passing each bisecting curve exactly once is given called Hamiltonian abstract Voronoi diagrams, and proposed a linear-time algorithm.

However, the convex position is not applicable for many other geometric objects and other distance measures. Furthermore, computing a Hamiltonian path is a still unsolved problem, and its existence is still unknown.

Therefore, we consider the abstract Voronoi diagram in a domain where its structure is a forest and each of its sites has exactly one face. Let \( D \subseteq \mathbb{R}^2 \) be a bounded domain, e.g. a domain bounded by \( \Gamma \), where \( \Gamma \) is a simple closed curve intersecting each bisector exactly twice such that no two bisectors intersect in a connected component entirely enclosed by the outer domain of \( \Gamma \). In the following, without explicit indication, \( V(S') \) means \( V(S') \cap D \) and \( \text{VR}(p,S') \) means \( \text{VR}(p,S') \cap D \).

Definition 1 For each set \( S' \subseteq S \) of sites let \( \pi(S') \) be the sequence of regions of \( V(S') \) along \( \partial D \). Since \( V(S) \) partitions \( \partial D \) into \( |\pi(S)| \) pieces, each element of \( \pi(S) \) corresponds to a unique piece. For each element \( p \) of \( \pi(S) \), \( d(p) \) is a point on its corresponding piece.

Remark that \( \text{VR}(p,S) \subseteq \text{VR}(p,S') \), thus \( d(p) \in \text{VR}(p,S') \) for all subsets \( S' \) of \( S \). We require the additional axiom:

(A4) \( V(S) \) is a tree. For all \( S' \subseteq S \), \( V(S') \) is a forest and each Voronoi region has exactly one face.

This axiom implies that each element in \( \pi(S) \) occurs only once. For subsets \( S' \) of \( S \) we have the following observation.

Lemma 1 \( \pi(S') \) is a Davenport-Schinzel-Sequence of order 2.

---

*This work was supported by the European Science Foundation (ESF) in the EUROCORES collaborative research project EuroGIGA/VORONOII.

This is an extended abstract of a presentation given at EuroCG 2014. It has been made public for the benefit of the community and should be considered a preprint rather than a formally reviewed paper. Thus, this work is expected to appear in a conference with formal proceedings and/or in a journal.
Actually, \( \pi(S) \) depends on the starting point and the direction of a traversal along \( \partial D \). Without loss of generality, we assume the starting point is known and the direction is \textit{clockwise}. Based on these assumptions and our axioms we prove the following result.

**Theorem 2** Given a domain \( D \) together with a sequence \( \pi(S) \) we can compute \( V(S) \) in time \( \mathcal{O}(n) \).

Compared to the algorithm for Hamiltonian abstract Voronoi diagrams in [5], our algorithm has two major differences in the coloring (Section 2.1) and selection (Section 2.2), and we prove the corresponding theoretical properties for the correctness. The coloring, our algorithm needs to consider two more sub-cases, and two consecutive sites in the sequence can be both colored red, while no consecutive sites in [5] are colored red. For the selection, our algorithm needs to modify \( V(S) \) into a tree for applying Aggarwal’s selecting lemma [1]. Note that finding a Hamiltonian path for the algorithm in [5] is still unknown and may be very hard.

## 2 The Algorithm

We want to use a recursive algorithm to compute \( V(S) \). To be able to recursively compute \( V(S') \) from \( V(S) \) it is important that the input, the sequence of sites \( \pi(S') \), fulfills the same properties as the sequence \( \pi(S) \). But \( \pi(S) \) is a \textit{Davenport-Schinzel-Sequence} (DSS) of order 1, whereas \( \pi(S') \) may be a DSS of order 2. For this purpose we will use the following definition.

**Definition 2** Let \( \pi'(S') \) be the subsequence of \( \pi(S) \) containing all elements from \( S' \), i.e. \( \pi'(S') \) is a DSS of order 1.

In the following we will see that it indeed suffices to consider the subsequence \( \pi'(S') \) in order to compute \( V(S') \). Now our algorithm can be summarized as follows:

1. Color each element of \( \pi(S) \) either \textit{blue} or \textit{red}, i.e., \( \pi \) is partitioned into \( \pi'(B) \) and \( \pi'(R) \), and \( S \) is partitioned into \( B \) and \( R \), such that both \( |B| \) and \( |R| \) are a constant fraction of \( |S| \), and for each two consecutive red sites, \( r_1 \) and \( r_2 \), in \( \pi \), VR(\( r_1 \)), \( B \cup \{ r_1, r_2 \} \) and VR(\( r_2 \)), \( B \cup \{ r_1, r_2 \} \) are not adjacent. See Section 2.1 for details.

2. Compute \( V(B) \) from \( \pi'(B) \) recursively.

3. Select a subset \( C \) from \( R \) such that \( |C| \) is a constant fraction of \( |R| \), and for any two sites, \( c_1 \) and \( c_2 \), VR(\( c_1 \)), \( B \cup \{ c_1, c_2 \} \) and VR(\( c_2 \)), \( B \cup \{ c_1, c_2 \} \) are not adjacent. See Section 2.2 for details.

   - Apply Aggarwal et al.’s selecting Lemma [1] on \( V^*(B) \)

   4. Compute \( V(B \cup C) \) by sequentially inserting each element of \( C \) into \( V(B) \).

   5. Compute \( V(G) \) from \( \pi'(G) \) recursively, where \( G = R \setminus C \) and \( \pi'(G) \) is obtained from \( \pi'(R) \) by removing all elements in \( C \).

   6. Merge \( V(B \cup C) \) and \( V(G) \).

Step 1 can be carried out in linear time according to Section 2.1, Step 3 and Step 4 can be completed in linear time according to Section 2.2 and 2.3, and Step 6 can be implemented in linear time using the general merge method described in [3]. Since \( |B| \) and \( |G| \) is a constant fraction of \( |S| \), the above claims conclude Theorem 2.

**Definition 3** For a set \( S \) of sites, a subset \( S' \) of \( S \), and a site \( p \) of \( S' \), a connected intersection between VR(\( p, S' \)) and \( \partial D \) is redundant if it does not contain the connected intersection between VR(\( p, S \)) and \( \partial D \).

From the viewpoint of \( \pi'(S') \), which is a subsequence of \( \pi(S) \), a connected intersection between VR(\( p, S' \)) and \( \partial D \) is redundant if it does not contain d(\( p \)) for \( p \) in \( \pi'(S') \).

**Definition 4** For all \( S' \subseteq S \), a \( pqr \)-vertex of \( V(S') \) is a Voronoi vertex adjacent to VR(\( p, S' \)), VR(\( q, S' \)), and VR(\( r, S' \)) clockwise. If VR(\( p, S' \)) is the only region bordering VR(\( q, S' \)), VR(\( p, S' \)) encloses VR(\( q, S' \)), for brevity we say \( p \) encloses \( q \) in \( V(S') \).

### 2.1 Red-Blue-Coloring Scheme

The coloring scheme consists of two steps, where a site is blue as long as it is not colored red. See also Figure 1.

1. For every 5 consecutive sites along \( \pi(S) \), \( l, m, p, q, r \), \( p \) is colored red if one of the following conditions holds. Let \( T = \{ l, m, p, q, r \} \).
   - (i) There is a \( mpq \)-vertex in \( V(T) \).
   - (ii) VR(\( m, T \)) encloses VR(\( p, T \)).
   - (iii) VR(\( q, T \)) encloses VR(\( p, T \)).

2. For every 3 consecutive sites along \( \pi(S) \) that are all blue, the middle one is colored red.

Let \( R \) be the set of red sites, and \( B \) be the set of blue sites. Observe that the final diagram \( V(S) \) is a tree, but in the recursion \( V(S) \) may be a forest, e.g. when \( S = B \). Then we use the sequence \( \pi'(S) \) instead of \( \pi(S) \).

**Lemma 3** No 3 consecutive sites in \( \pi(S) \) are all colored red.
Proof. For the sake of a contradiction assume that three consecutive sites $r_1, r_2, r_3$ are all red. Let $s_1$ and $s_2$ be the two consecutive sites previous to $r_1$, and $s_3$ and $s_4$ the two consecutive sites after $r_3$. By definition $r_1, r_2$ and $r_3$ can not be colored red by step 2. Thus we need only consider step 1.

There are three different cases for $r_1$ to be colored red.

Case 1: There is an $s_2 r_1 r_2$-vertex in $V(\{s_1, s_2, r_1, r_2, r_3\})$. This vertex is still an $s_2 r_1 r_2$-vertex in $V(\{s_2, r_1, r_2, r_3\})$ implying that there can not exist an $r_1 r_2 r_3$-vertex in $V(\{s_2, r_1, r_2, r_3\})$ and hence also no $r_1 r_2 r_3$-vertex in $V(\{s_2, r_1, r_2, r_3, s_3\})$. This means that $r_2$ must be colored red because $r_1$ or $r_3$ encloses it in $V(\{s_2, r_1, r_2, r_3, s_3\})$. But then $r_2$ can not be adjacent to a vertex in $V(\{s_2, r_1, r_2, r_3\})$, i.e., no $s_2 r_1 r_2$ vertex exists, a contradiction.

Case 2: $r_1$ is colored red because $s_2$ encloses $r_1$ in $V(\{s_1, s_2, r_1, r_2, r_3\})$. But then the regions of $r_1$ and $r_2$ are not adjacent in $V(\{s_2, r_1, r_2, r_3\})$ and there can be no $r_1 r_2 r_3$-vertex in $V(\{s_2, r_1, r_2, r_3, s_3\})$. Further $r_1$ can not enclose $r_2$. This means that $r_2$ must be colored red because $r_3$ encloses it in $V(\{s_2, r_1, r_2, r_3, s_3\})$. But then there can be no $r_2 r_3 s_3$-vertex in $V(\{r_1, r_2, r_3, s_3, s_4\})$ and $r_3$ can not be enclosed by $r_2$ or $s_3$ in $V(\{r_1, r_2, r_3, s_3, s_4\})$. Thus $r_3$ is not colored red, a contradiction.

Case 3: $r_1$ is enclosed by $r_2$ in $V(\{s_1, s_2, r_1, r_2, r_3\})$ but then because of the same reasons as in case two $r_2$ is not colored red.

Corollary 4 Let $s_1, s_2, r_1, r_2, s_3, s_4$ be 6 consecutive sites in $\pi(S)$. If $r_1$ and $r_2$ are both red, then $s_2$ and $s_3$ are both blue. Further $s_2$ encloses $r_1$ in $V(\{s_1, s_2, r_1, r_2, s_3\})$ and $s_3$ encloses $r_2$ in $V(\{s_2, r_1, r_2, s_3, s_4\})$. In particular $s_2$ encloses $r_1$ and $s_3$ encloses $r_2$ in $V(\{s_2, r_1, r_2, s_3\})$.

It can happen that two consecutive sites are both colored red, see Figure 2 However, we still have the following property.

Lemma 5 Let $r_1$ and $r_2$ be two consecutive red sites. Then $\text{VR}(r_1, B \cup \{r_1, r_2\})$ and $\text{VR}(r_2, B \cup \{r_1, r_2\})$ are not adjacent.

Proof. (Sketch) According to the coloring method, there are three cases: (1) no blue site, (2) one blue site, and (3) two blue sites between $r_1$ and $r_2$. Corollary 4 implies case 1, and case 2 and 3 can be proven by contradiction in which a blue site should be colored red. □

2.2 Choosing Crimson Sites

We want to apply the following combinatorial lemma from [1] to obtain an independent set of crimson sites.

Lemma 6 Let $T$ be a binary tree embedded in the plane and for each leaf $l$ a subtree $T_l$ rooted at $l$ is defined. Further the subtrees of two consecutive leaves in the topological ordering around $T$ are disjoint. Then one can in linear time find a fixed fraction of leaves whose subtrees are pairwise disjoint.

To use this lemma we will modify the forest $V(B)$, generated by the blue sites, by adding some edges and leaves to obtain a tree $V^*(B)$ fulfilling the claimed properties. We start with the following observation.

Lemma 7 We can detect all redundant intersections of $V(B)$ in time $O(n)$.

Now we construct $V^*(B)$ out of $V(B)$ by the following operations, compare Figure 3.

(i) For all redundant intersections on $\partial D$ link the two leaves bounding it along $\partial D$.

If the redundant intersection borders another redundant intersection on its right end, then let the leaf between them now be a vertex in $V^*(B)$. Observe that this is a vertex of degree 3. Otherwise connect the right end of the link to $V(B)$ without creating a vertex. The same is done on the left side of the redundant intersection.

Next we attach some leaves to $V^*(B)$ outside of $D$ such that between each pair of consecutive blue sites $b_i$ and $b_{i+1}$ having one (or two) red site(s) in between, there is exactly one (or between one and two) leaves in $V^*(B)$. If there is no red site between $b_i$ and $b_{i+1}$ there is also no leaf.

(ii) If there are one or two red sites $r_1$ and $r_2$ between two consecutive blue sites $b_i$ and $b_{i+1}$ but no leaf between them, then there is a connected set of redundant intersections between $b_i$ and $b_{i+1}$. If $d(r_j), j = 1, 2$ lies within this sequence we attach a leaf to $V^*(B)$ at $d(r_j)$, otherwise if $d(r_j)$ lies to
the left (right) of the sequence we attach a leaf at the leftmost (rightmost) point of the sequence. If both \(d(r_j), j = 1, 2\) are to the left (right) of the redundant intersection sequence, only one leaf is attached at the leftmost (rightmost) point.

(iii) If there is a leaf in \(V(B)\) between two consecutive blue sites \(b_i\) and \(b_{i+1}\), which are not separated by a red site, it is pruned like in [5].

\[
\pi(S) = (k, l, m, p, q, r, s, t, u, v, w, x, y, \ldots) \\
\pi'(B) = (l, p, r, t, u, x, y, \ldots) \\
\pi(B) = (t, l, p, r, t, u, t, y, x, y, \ldots)
\]

Figure 3: \(V^*(B)\), fat edges indicate redundant intersections and new leaves.

Lemma 8 \(V^*(B)\) is a binary tree and can be constructed in time \(O(n)\).

Proof. (Sketch) It is clear that \(V^*(B)\) is a forest. So assume it is disconnected. Then there is a site \(b \in B\) whose Voronoi region in \(V(B)\) intersects \(\partial D\) in more than one component. By (i) all these components are non redundant. But then \(b\) has to appear several times in \(\pi(S)\), a contradiction. Definitions (i) to (iii) imply that all internal nodes of \(V^*(B)\) are of degree 3.

By lemma 7 we can detect all redundant intersections in time \(O(n)\). In the same time operation (i) can be accomplished. For operation (ii) and (iii) we have to walk one time around \(\partial D\), compute the number of red sites between each pair of consecutive blue sites, detect the locations of \(d(r)\) for the red sites, attach leaves like described in (ii) and prune others like described in (iii). For each redundant intersection this takes constant time and there are \(O(n)\) redundant intersections altogether.

Coloring Crimson:

If two blue sites \(b_i\) and \(b_{i+1}\) are separated by a red site \(r\) in \(\pi(S)\) but the leaf between them is not contained in \(VR(r, B \cup \{r\})\), then \(r\) is enclosed by the region of \(b_i\) (or \(b_{i+1}\)) in \(V(B \cup \{r\})\). In this case color \(r\) crimson with respect to \(b_i\) (or \(b_{i+1}\)) and if the leftmost (rightmost) leaf between \(b_i\) and \(b_{i+1}\) is not contained in the region of a consecutive site, associate with \(r\) the subtree containing only this leaf. If two red sites are between \(b_i\) and \(b_{i+1}\) and both are colored crimson because of \(b_i\) (or \(b_{i+1}\)) associate only one of them with the leftmost (rightmost) leaf.

Up to now we may already have colored some red sites crimson. To make sure we receive a fixed fraction of crimson sites we apply lemma 6 in the following way. For each leaf \(l\) of \(V^*(B)\) contained in a red region \(VR(r, B \cup \{r\})\) define \(T_i\) by the subtree spanned by all vertices of \(V^*(B)\) contained in \(VR(r, B \cup \{r\})\). The next lemma shows that this is possible.

Lemma 9 Let \(r\) be a red site. If \(VR(r, B \cup \{r\})\) intersects a leaf of \(V^*(B)\), then \(VR(r, B \cup \{r\}) \cap V^*(B)\) is connected. Otherwise it is empty.

Proof. Suppose \(VR(r, B \cup \{r\})\) intersects a leaf of \(V^*(B)\) and \(VR(r, B \cup \{r\}) \cap V^*(B)\) is not connected. Then \(VR(r, B \cup \{r\})\) would disconnect the region of a blue site in \(V(B \cup \{r\})\), a contradiction.

If \(VR(r, B \cup \{r\})\) does not intersect a leaf of \(V^*(B)\) it must be contained within a single blue region of \(V(B)\), thus it can not intersect \(V^*(B)\).

Now Lemma 5 and 6 imply the requested property.

Lemma 10 No regions of two crimson sites are adjacent in \(V(B \cup C)\).

2.3 Insertion of Crimson Sites

We can now insert the crimson sites into \(V(B)\) in order to receive \(V(B \cup C)\). For each crimson site \(c\) whose region does not intersect a leaf of \(V^*(B)\) we know that it is enclosed by the region of a blue site \(b_i\) or \(b_{i+1}\). Let it be \(b_i\), then we just have to insert the part of the bisector \(J(r, b_i)\) contained in \(VR(b_i, B)\) as a new edge in \(V(B \cup C)\). The other crimson sites can be inserted along the subtree of \(V^*(B)\) associated with them. Thus also the insertion takes time \(O(n)\).

References