Rolf Klein, Andrzej Lingas

Hamiltonian Abstract Voronoi Diagrams in Linear Time

Mathematik und Informatik

Informatik-Berichte
161 – 05/1994

Hamiltonian Abstract Voronoi Diagrams in Linear Time

Rolf Klein * Andrzej Lingas †

May 11, 1994

Abstract

Let $V(S)$ be an abstract Voronoi diagram, and let H be an unbounded simple curve that visits each of its regions exactly once. Suppose that each bisector $B(p,q)$, where p and q are in S, intersects H only once. We show that such a "Hamiltonian" diagram $V(S)$ can be constructed in linear time, given the order of Voronoi regions of $V(S)$ along H. This result generalizes the linear time algorithm for the Voronoi diagram of the vertices of a convex polygon. We also provide, for any $\delta > \log_{60/29} 2$, an $O(n^\delta)$-time parallelization of the construction of the $V(S)$ optimal in the time-processor product sense.

Key words: Computational geometry, Voronoi diagrams, abstract Voronoi diagrams, convex polygons

1 Introduction

The Voronoi diagram of a set S of n sites is one of the most useful structures in computational geometry. It partitions the plane into regions, one to each site p in S containing all points that are closer to p than to any other site in S. Shamos and Hoey [15] proved the construction of the Euclidean Voronoi diagram to be of complexity $\Theta(n \log n)$, giving a divide-and-conquer algorithm. Later, Fortune [9] introduced an $O(n \log n)$ sweep line algorithm,

*FernUniversität Hagen, Praktische Informatik VI, Elberfelder Straße 95, 58084 Hagen. e-mail: rolf.klein@fernuni-hagen.de. This work was partially supported by the Deutsche Forschungsgemeinschaft, grant Kl 655/1-2. It was begun during the first author's visit at Lund University in March 1993.

†Lund University, Department of Computer Science. e-mail: andrzej@dna.lth.se.

and Clarkson and Shor [3] presented a randomized incremental construction technique.

Voronoi diagrams can be generalized in many ways. Unifying concepts have been suggested by Edelsbrunner and Seidel [8] and Klein [11]. In the latter approach, the definition "point x is closer to site p than to site q" is replaced by "point x lies on the p-side of the bisector, $B(p,q)$, of p and q", this way abstracting from the concepts of both sites (as physical objects) and distance. For the resulting class of *abstract* Voronoi diagrams an optimal divide-and-conquer algorithm was given in [11], that works if the bisector of the two sets of sites generated by the divide step does not contain loops. An $O(n \log n)$ randomized algorithm that works without such assumptions has been presented in Klein, Mehlhorn, and Meiser [12].

It is very natural to ask whether Voronoi diagrams can be computed faster if more information about the position of the sites is available. A famous open problem mentioned by Preparata and Shamos [14] addresses the vertices of a convex polygon, given in cyclic order. It has been solved by Aggarwal, Guibas, Saxe, and Shor [1] by providing an $O(n)$ algorithm. Their algorithm first applies the geometric lifting mapping suggested in [8] to a paraboloid in 3D, and then constructs the convex hull of the transformed points. The dual of their (lower) convex hull gives the desired Voronoi diagram, after taking the projection to the plane.

This transformation makes the algorithm technically somewhat complicated. Moreover, it clouds the fact that the kernel of this powerful method is not really based on geometric properties, as suggested by the convexity assumption, but only on combinatorial properties of the underlying bisector system, as we show in this paper.

By working directly in the plane we are able to cover situations more general than convex point sets. All we need to assume is the following. The sites are given in the order in which their regions in $V(S)$ are visited by an unbounded simple curve H that passes through each region exactly once. Also, for each subset S' of S, curve H is assumed to visit each region of $V(S')$ not more often than once. In presence of the former condition, the latter is equivalent to saying that each bisector $B(p,q)$, where $p, q \in S$, crosses H only once; see Lemma 3.1. We call a such a Voronoi diagram *Hamiltonian with respect to* H. On either side of H, its structure is that of a forest of binary trees (and possibly some halflines). We show that a Hamiltonian diagram can be computed in linear time, given the order of regions along H.

There is an easy way of creating examples of Hamiltonian diagrams. Let H be an unbounded curve such that for any two points p, q on H their Euclidean bisector $B(p, q)$ crosses H only once. Then the Voronoi diagram of any finite set of points on H is Hamiltonian with respect to H. This includes convex curves [1], monotone histograms, as studied by Djidjev and Lingas [6], and convex chains of straight and circular segments TO BE CHECKED investigated by Yap [16], thereby generalizing previous results.

Our result is not restricted to the Euclidean metric. Since it holds true for abstract Voronoi diagrams it also works for all concrete metrics covered by this concept, e. g. the family of "nice" metrics described in [11]. This family includes all semi-algebraic convex distance functions.

The problem of whether there exist fast parallel algorithms for the aforementioned special site configurations more work efficient than the known parallel algorithms for the general configurations (e.g., see [5]) is also widely open. We are able to parallelize all the basic steps in the linear-time algorithm for the Hamiltonian diagrams (or in the algorithm of Aggarwal *et al*). However the presence of the second recursive call which has to wait for the outcome of the first one slows down the time performance of our parallelization to vaguely sublinear. In effect, we obtain the following generalization of our linear-time bound for the Hamiltonian diagrams: For any $\delta > \log_{60/29} 2$, the Voronoi diagram Hamiltonian with respect to H can be computed in time $O(n^{\delta})$ using $O(n^{1-\delta})$ processors in the CREW PRAM model.

The paper is organized as follows. After briefly introducing abstract Voronoi diagrams in Section 2, we prove the main result in Section 3. In Section 4, we provide the parallelization. Finally, in Section 5 we discuss examples of Hamiltonian Voronoi diagrams.

2 Abstract Voronoi diagrams

Abstract Voronoi diagrams are designed to match what most concrete Voronoi diagrams have in common. They are defined by means of a set S of n indices $p, q, \ldots$ that play the role of sites. For any subset $\{p, q\}$, where $p \neq q$, there is a simple unbounded curve $B(p, q)$, called *bisector*, that partitions the plane into two unbounded open domains, $D(p, q)$ and $D(q, p)$. For technical reasons, we require that bisectors are homeomorphic to lines, and that any two of them intersect in only finitely many connected components. Voronoi regions can now be defined by analogy to intersecting halfplanes.

Definition 2.1 Let S and the family $\{B(p,q); p,q \in S, p \neq q\}$ as before. Then

$$VR(p,S) = \bigcap_{q \in S, p \neq q} D(p,q) \text{ and}$$

$$V(S) = \bigcup_{p,q \in S, p \neq q} \overline{VR}(p,S) \cap \overline{VR}(q,S)$$

are called the *Voronoi region of p with respect to S* and the *Voronoi diagram* of S, correspondingly.

This definition is too general for describing standard Voronoi diagrams. Therefore, the following conditions are imposed.

Definition 2.2 The family $\{B(p,q); p,q \in S, p \neq q\}$ is called *admissible* if for each subset S' of S, where $|S'| \geq 3$, the following hold.

1. Each region $VR(p,S')$ is connected.
2. Each point of the plane either belongs to a Voronoi region $VR(p,S')$ or to the Voronoi diagram $V(S')$.

Definition 2.1 is due to Meiser [13]; it is more convenient than the original one given in [11]. The above properties are sufficient for deriving a structure theory, and for designing efficient algorithms; see [11, 12]. For simplicity, let us assume that any two bisectors $B(p,q)$ and $B(q,r)$ of three sites cross transversally whenever they intersect. Then the following properties are fulfilled that will be needed later.

Lemma 2.3 *1. If $B(p,q)$ and $B(q,r)$ cross at point x then $B(p,r)$ also passes through x. Locally at x, the Voronoi diagram $V(\{p,q,r\})$ has the same structure as a Euclidean diagram.*

2. $B(p,q)$ and $B(q,r)$ cross at most twice. If so, the bounded part of the plane they enclose belongs to at least one of $D(p,q)$ or $D(p,r)$.

3. Voronoi regions do not have holes, i. e. they are simply-connected.

See Figure 1 for an illustration and [11] for a proof.

All semi-algebraic convex distance functions generate Voronoi diagrams that comply with Definitions 2.1 and 2.2. But many other concrete diagram types are also covered, e. g. the Euclidean Voronoi diagram of line segments, where two bisectors associated with the same line segment can in fact cross twice.

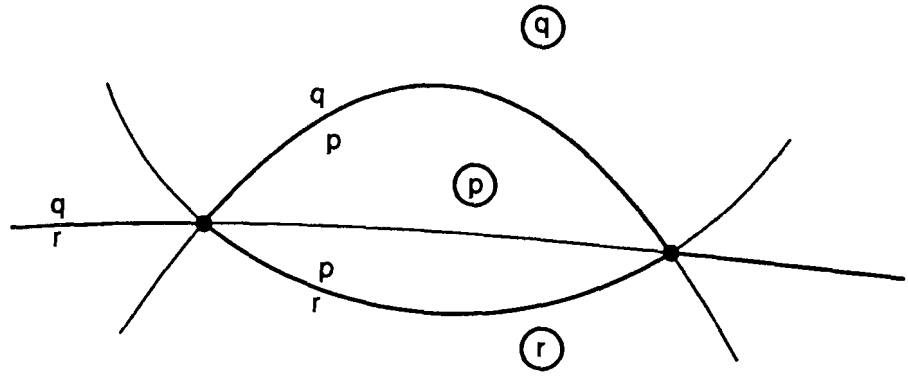

Figure 1: Two bisectors $B(p,q)$ and $B(q,r)$ cross at most twice.

3 Constructing Hamiltonian diagrams

Throughout this section, let $\{B(p,q); p,q \in S, p \neq q\}$ be an admissible bisector system, and $n = |S|$. Moroeover, let H denote a simple, unbounded curve, homeomorphic to a line, that passes exactly once through each region of the abstract Voronoi diagram $V(S)$. Necessarily, the first and the last regions visited by H must be unbounded.

Lemma 3.1 *The following assertions are equivalent.*

1. *For each subset S' of S, $|S'| \geq 2$, each region of $V(S')$ is also visited by H exactly once.*

2. *Each bisector $B(p,q)$, where $p,q \in S$, crosses H exactly once.*

Proof: 1) $\Rightarrow$ 2) : If $B(p,q)$ crossed H more than once then H would visit the regions of either p or q in $V(\{p,q\})$ more than once. If $B(p,q)$ and H were disjoint then one of the regions of p,q would not be visited at all.
2) $\Rightarrow$ 1) : For each site $p \in S'$, its region can only grow as we change from S to S', so it will still be visited by H. Assume it is visited more than once, and H passes through the region of some other site $q \in S'$ in between. Then the same holds true in $V(\{p,q\})$, showing that $B(p,q)$ crosses H at least twice. □

Definition 3.2 If $\{B(p,q); p,q \in S, p \neq q\}$ and H fulfill the properties stated in Lemma 3.1 then $V(S)$ is called *Hamiltonian with respect to* H.

Now we state the main result.

Theorem 3.3 *Let $V(S)$ be Hamiltonian with respect to H. Assume that we are given the sites of S in the order in which their regions are visited by H. Then $V(S)$ can be computed in time $O(n)$.*

Curve H partitions the plane into two unbounded domains. To prove the theorem we need only show that the part of $V(S)$ lying in one of them, D, can be constructed in linear time. For simplicity, let us assume that there are no Voronoi vertices of degree greater than 3 (the algorithm can be made to run without this assumption.)

Lemma 3.4 *$V(S) \cap D$ is a forest of halflines and binary trees with one edge extending to infinity and leaves on H.*

Proof: Let the regions of p, q, r, s be consecutive on H. Between the regions of q and r, a Voronoi edge originates from H that is part of the bisector $B(q,r)$. Either this edge extends to infinity, giving rise to a halfline in $V(S) \cap D$, or it crosses one of its two neighbors, say $B(p,q)$. At the cross point, a new edge belonging to $B(p,r)$ originates, due to Lemma 2.3, 1). After removing site q from S this argument can be iterated. □

The basic idea of the construction of $V(S) \cap D$ is quite simple. In the following discussion we are using the coloring scheme from [1], to indicate the relation to the transformed dual objects the algorithm presented there works with.

Algorithm 1

Input: The order of sites in S in which their regions are visited by the curve H.
Output: The Voronoi diagram $V(S)$ within the halfplane D.

1. Choose at least a fixed percentage of sites whose regions in $V(S)$ are pairwise non-adjacent, and color *red* the sites in this subset, R, of S.
2. Color *blue* the remaining sites in $B = S - R$, and compute recursively their Voronoi diagram $V(B)$ within D.

3. Construct incrementally $V(S) \cap D$ from $V(B) \cap D$, inserting the red sites one by one.

Note that it is legal to apply this algorithm recursively in Step 2 since Lemma 3.1, 1) holds for the set B.

Lemma 3.5 *Step 3 can be accomplished within time $O(n)$.*

Proof: Since no two red regions share an edge in $V(S)$ we can insert the red sites independently, so it suffices to construct $V(B \cup \{r\})$ for each r in R.

We know the blue sites b_i, b_{i+1} whose regions are, on H, neighbors of the region of r. Thus, we can in constant time locate starting segments for the contour of $VR(r, B \cup \{r\})$ in the intervals on H covered by the regions of b_i and b_{i+1} in $V(B)$; see Figure 2.

Next we study the part of $V(B)$ contained in $VR(r, B \cup \{r\})$. Certainly, the region of r always contains an initial piece of $B(p_i, p_{i+1})$ originating from a blue leaf l on H, because this edge is no longer present in $V(B \cup \{r\})$. We claim that $T_l = V(B) \cap VR(r, B \cup \{r\})$ is connected. Namely, if the contour of $VR(r, B \cup \{r\})$ entered the same blue region twice, like the region of r_3 in Figure 2, this region would become disconnected in $V(B \cup \{r\})$—a contradiction to Definition 2.2. If T_l contains more than the blue edge originating from leaf l then it must be a binary tree. But l is the only point of T_l on the curve H.

Since T_l is connected we can trace the contour of $VR(r, B \cup \{r\})$ in time proportional to the number of edges of T_l, simply by walking around the tree (we may have to jump at infinity if the region of r is unbounded, like for r_2 in Figure 2).

For the blue edges completely contained in the new region of r in $V(B \cup \{r\})$ we charge this work to the edge itself. Each edge that is only partially covered by $VR(r, B \cup \{r\})$ gives rise to a Voronoi vertex of $V(S)$, which gets charged.

Therefore, the total cost of Step 3 is $O(n)$. □

Leaving aside Step 1 of Algorithm 1, we would obtain the recursion

$$T_1(n) \leq T_1(qn) + Cn,$$

where $q < 1$ bounds the maximal relative size of B. This shows that $T_1(n)$ is $O(n)$.

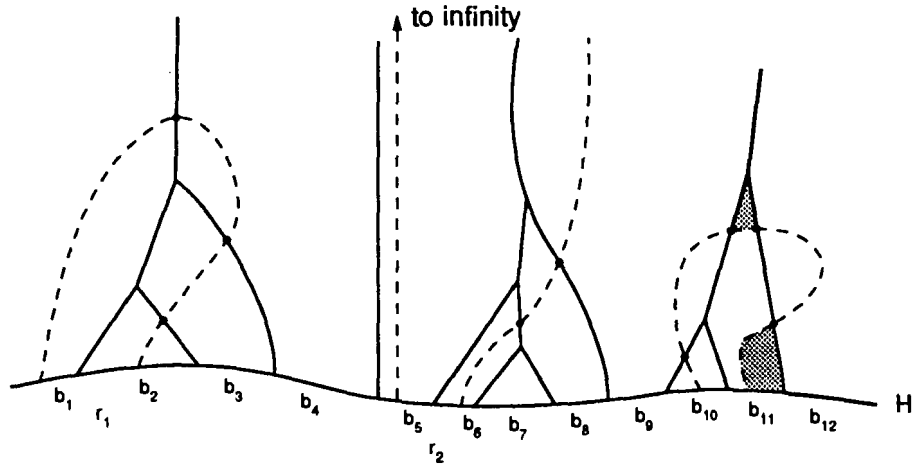

Figure 2: Inserting the red sites r_i. The region of r_3 cannot look as indicated, because the region of b_{11} would become disconnected.

The obvious problem is that Step 1 cannot be accomplished without knowing $V(S)$ first. Therefore, Algorithm 1 is modified in the following way. One first chooses a suitable *blue* subset B and computes $V(B)$. Then the knowledge of $V(B)$ is used for identifying a *crimson* subset C of $S-B$ whose elements do have independent Voronoi regions in $V(B \cup C)$, as required in Step 1 of Algorithm 1, so they can be inserted in linear time, as before. For the remaining *garnet* sites $G = S-(B \cup C)$, the diagram $V(G)$ is computed recursively, and merged with $V(B \cup C)$ into $V(S)$. This merge step can be implemented to run in time $O(n)$ using the general method described in [11], Theorem 3.4.3.2, because the bisector of the site sets $B \cup C$ and G does not contain loops.

In the following, only the parts of the Voronoi diagrams contained in domain D are considered, without further mention.

Algorithm 2

Input: The order of sites in S in which their regions are visited by the curve H.
Output: The Voronoi diagram $V(S)$.

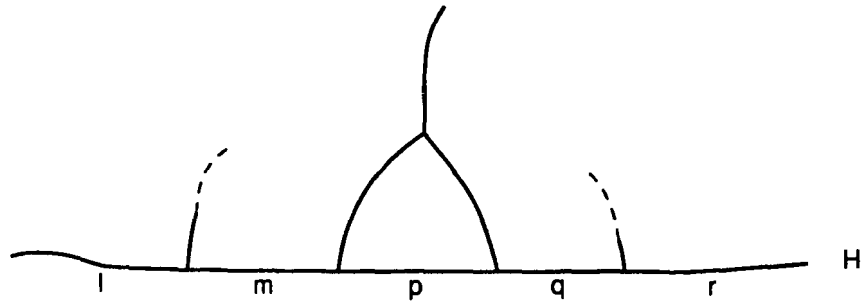

Figure 3: Site p is colored red iff its two bisectors cross in $V(\{l, m, p, q, r\})$.

1. Choose a suitable *blue* subset B of S.
2. Compute recursively the Voronoi diagram $V(B)$.
3. Using the structure of $V(B)$, choose a *crimson* subset C of $S - B$, such that no two regions of crimson sites are adjacent in $V(B \cup C)$.
4. Construct incrementally $V(B \cup C)$ from $V(B)$, inserting the crimson sites one by one.
5. Compute recursively the *garnet* Voronoi diagram $V(G)$, where $G = S - (B \cup C)$.
6. Merge $V(B \cup C)$ with $V(G)$ into $V(S)$.

Step 4 can be carried out in linear time, according to Lemma 3.5. Therefore, the running time of Algorithm 2 will be linear if we can ensure that in Step 3 at least a fixed percentage of sites becomes crimson, and that Steps 1 and 3 can be carried out in linear time.

Implementation of Step 1. Let l, m, p, q, r denote five sites of S whose regions on H are consecutive. Site p is colored *red* iff, in the Voronoi diagram $V(\{l, m, p, q, r\})$, the bisectors $B(m, p)$ and $B(p, q)$ cross, giving rise to a Voronoi vertex; see Figure 3.

Lemma 3.6 *So far, no two consecutive sites have been colored red.*

Proof: Suppose that in the sequence of consecutive sites l, m, p, q, r, s both p and q have been colored red. By definition, $B(p,q)$ makes a vertex with its left neighbor, $B(m,p)$, in $V(\{l,m,p,q,r\})$, and with its right neighbor, $B(q,r)$, in $V(\{m,p,q,r,s\})$. But only one of these facts can be true in $V(\{m,p,q,r\})$—a contradiction! □

In case no site has been colored red we know that $V(S)$ consists of halflines only, because any Voronoi vertex of $V(S)$ that is just above the leaf level in its binary tree (cf. Lemma 3.4) would also be present in one of the local diagrams of five sites. In this case, we can stop. Otherwise, there are at least two sequences of consecutive sites that have no color yet. Within each such sequence, every other site is colored red, in a way that no two consecutive sites become red. All remaining sites are colored *blue*; they form the subset B of S. Between any two successive red sites there are either one or two blue sites.

Clearly, all these local computations can be done in linear time because they involve only a linear number of constant size Voronoi diagrams.

Next, we list two properties of the blue and red sites that will be used in Step 3.

Lemma 3.7 *Let p, r be two successive red sites.*

1. *The regions of p and r in $V(B \cup \{p,r\})$ are not adjacent.*
2. *The regions of p and r in $V(B \cup \{p\})$ and $V(B \cup \{r\})$, correspondingly, are disjoint.*

Proof: 1) Suppose there is *one* blue vertex, q, between p and r. If the contours of $VR(p, B \cup \{p,r\})$ and $VR(r, B \cup \{p,r\})$ have a piece of $B(p,r)$ in common then $B(p,q)$ must meet $B(q,r)$ at a Voronoi vertex. Since both the predecessor of p and the successor of r are in B (p and r being red!) this Voronoi vertex also exists in the local diagram of these five sites. But then q should have been colored red at the beginning.

In case there are *two* blue sites, q_1 and q_2, between p and r, the bisector of q_1 and q_2 must produce a vertex with either $B(p,q_1)$ or $B(q_2,r)$, so that q_1 or q_2 should be red.

2) Assume that $I = VR(p, B \cup \{p\}) \cap VR(r, B \cup \{r\})$ is non-empty. If $VR(p, B \cup \{p\})$ were fully contained in $VR(r, B \cup \{r\})$, so would be $VR(p, B \cup \{p,r\})$. But this is impossible because on curve H the regions of p and r are

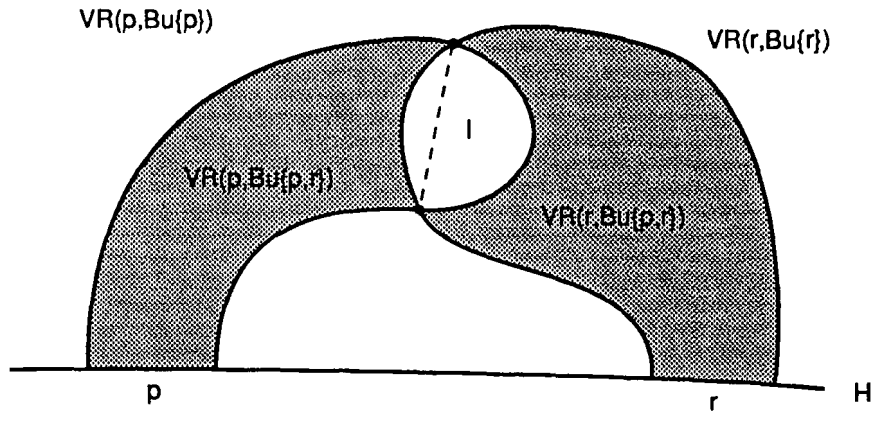

Figure 4: In $V(r, B \cup \{p, r\})$ there must be a Voronoi edge in I separating the regions of p and r.

separated by a blue region. Therefore, I is properly contained in either constituent set, as shown in Figure 4. Since the dashed area $VR(p, B \cup \{p\}) - I$ must be contained in the region of p with respect to $B \cup \{p, r\}$, and the same for r, these regions must have a common border inside I, contradicting 1). □

Implementation of Step 3. Lemma 3.7 does not exclude that two *non*-successive red regions may be adjacent. However, one can find an independent subset using the following combinatorial lemma from [1].

Lemma 3.8 *There exists a constant $q < 1$ such that the following holds. Let T be an unrooted binary tree, embedded in the plane. Assume that for each leaf l of T a subtree T_l rooted at l is defined, such that*

1. *given leaf l, one can in constant time decide if its parent node belongs to T_l,*
2. *if l, l' are consecutive leaves in the topological ordering around T then T_l and $T_{l'}$ are disjoint.*

Then it is possibel to find, in time $O(|T|)$, at least $q|T|$ many leaves whose subtrees have a pairwise edge distance greater than one.

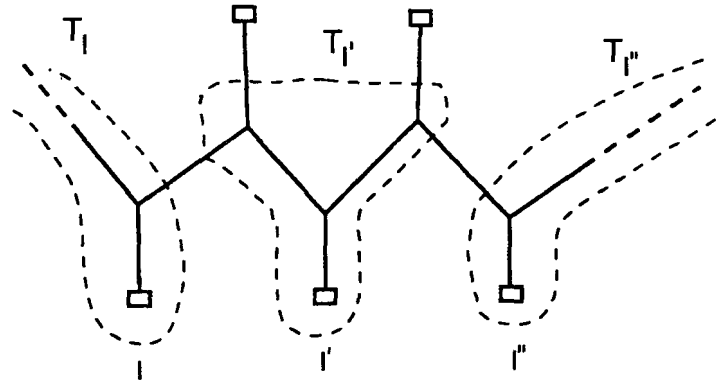

Figure 5: Selecting an isolated subtree in a group of leaves.

Here, the edge distance between two subtrees $T_l, T_{l'}$ denotes the minimum number of edges needed to connect a node of T_l to a node of $T_{l'}$ by a path in T. A positive edge distance in particular implies that the subtrees are disjoint.

Proof: (Sketch) If two consecutive leaves of T have the same parent node then one of them must have a subtree consisting only of the leaf itself, due to the disjointness assumption. All of these leaves are chosen. Besides such twin leafes, there can be groups of five leaves dangling off a spine, as shown in Figure 5. At least three of them, l, l', l'' in the picture, are consecutive. If one of l, l'' has a trivial subtree, we choose it. If both T_l and $T_{l''}$ are of size greater than one, then $T_{l'}$ is confined to the area indicated, and l' is chosen. □

This lemma is now applied to the forest $V(B)$ in the following way. For each pair of consecutive blue sites b_i, b_{i+1} that are *not* separated by a red site we conceptually remove the piece of $B(b_i, b_{i+1})$ that separates them in $V(B)$. If this piece has not been a halfline, we join the two remaining Voronoi edges incident to its parent vertex into one; see Figure 6.

As a result of this pruning step, there is a one-to-one relation between the leaves of the remaining forest, T, and the red sites. Namely, each leaf l of T lies in the region $VR(r, B \cup \{r\})$ of a unique site $r \in R$. We as-

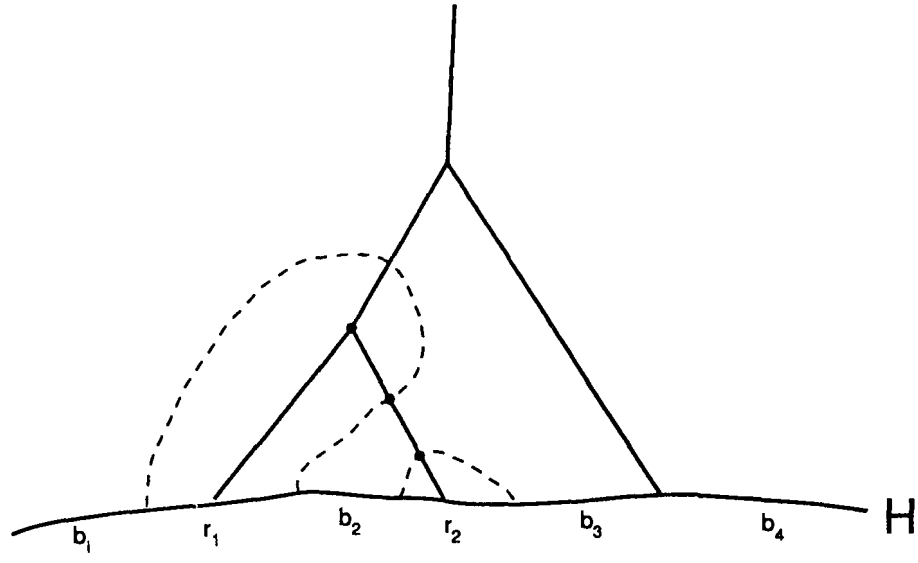

Figure 6: Blue leaves not contained in a red region are pruned. With the others, a unique red site is associated.

sociate with l the subtree T_l of T spanned by all vertices of T contained in $VR(r, B \cup \{r\})$; compare the proof of Lemma 3.5 and Figure 6. Now Lemma 3.7 2) guarantees the disjointness of consecutive subtrees. ¿From Lemma 3.8 we obtain, in linear time, a subset of leaves of T whose associated red sites $r, r', \ldots$ have the following property: no two of the regions $VR(r, B \cup \{r\})$, $VR(r', B \cup \{r'\}), \ldots$ intersect the same blue edge of T. These red sites $r, r', \ldots$ are now colored *crimson*. They form a subset C of S whose cardinality is at least a fixed percentage of $|S|$.

Lemma 3.9 *The regions of no two crimson sites are adjacent in* $V(B \cup C)$.

Proof: Let c, c' be crimson. We first show that the intersection I of their regions $VR(c, B \cup \{c\})$, $VR(c', B \cup \{c'\})$ is empty. Suppose this is not true. By the above, I does not contain any piece of T. Let e be one of the edges of $V(B)$ that have been pruned because its endpoint on H does not lie in any red region. Its other endpoint could be contained in the region of e. g. c. But then e cannot pass through $VR(c', B \cup \{c'\})$, because $V(B) \cup VR(c', B \cup \{c'\})$

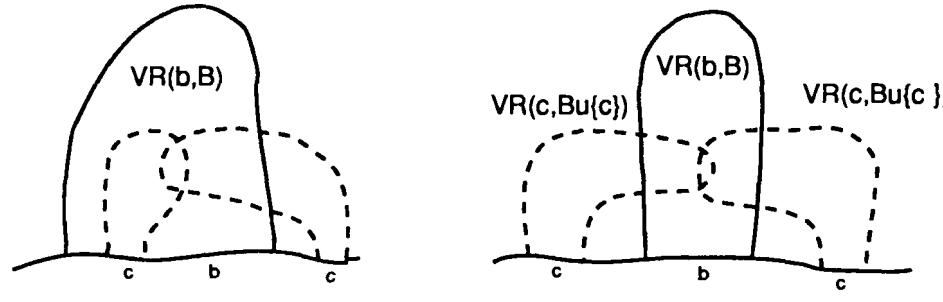

Figure 7: Neither situation is possible. In the left picture, the region of b in $V(B \cup \{c,\})$ would occur twice on H. In the right picture, the region of b would become disconnected in the diagram of $B \cup \{c, c'\}$.

is a tree one of whose edges originates from H. Therefore, e cannot pass through I either.

This implies that I is contained in some blue region $VR(b, B)$. On curve H, neither $VR(c, B \cup \{c\})$ nor $VR(c', B \cup \{c'\})$ can be fully contained in $VR(b, B)$. Therefore, both regions invade $VR(b, B)$ from the outside; see Figure 7. But then $VR(b, B)$ is disconnected in $V(B \cup \{c, c'\})$—a contradiction to Definition 2.2. The same contradiction arises if I is empty but the two regions touch each other. □

This completes the proof of Theorem 3.3.

Naturally, there is also a cyclic version of Theorem 3.3 which resembles the result in [1].

Theorem 3.10 *Let $\{B(p, q); p, q \in S, p \neq q\}$ be an admissible system, and suppose that the subdiagram of any three sites in S contains exactly one Voronoi vertex. If $V(S)$ is known to be a tree, and if the cyclic order of regions at infinity is given, then $V(S)$ can be computed in linear time.*

Proof: Analogous to the above. One has to make sure that, for each subset S' of S, $V(S')$ is still a tree, in order to apply recursion. In fact, it could happen that, after removing one site from S, another one "brakes through" to the other side, so that the diagram is no longer connected. But then there

would be three sites whose diagram is not connected either, contradicting the assumption. □

4 Parallel construction of Hamiltonian diagrams

The presence of the second recursive call waiting for the outcome of the first recursive call diminishes the possibility of a direct NC parallelization of Algorithm 2. In this section we present a vaguely sublinear-time parallelization of Algorithm 2 in the CREW PRAM model. In effect, we obtain the following generalization of our main result given in Theorem 3.3.

Theorem 4.1 *Let $V(S)$ be Hamiltonian with respect to H. Assume that we are given the sites of S in the order in which their regions are visited by H. Then, for any $\delta > \log_{60/29} 2$, $V(S)$ can be computed in time $O(n^\delta)$ using $O(n^{1-\delta})$ processors in the CREW PRAM model.*

Proof: Suppose that all the operations listed in the specification of Algorithm 2 can be computed in $t(n)$ time using $O(n/t(n))$ processors in the CREW PRAM model. Note that at least half of the sites is colored blue and at least one third of them is colored red (the minimum number of red sites is achieved when the primary red sites divide the site sequence into pairs of uncolored consecutive sites). Also, by the proof of the combinatorial lemma from [1] the constant q can be set to $1/10$ in Lemma 3.8. It follows that the crimson set includes at least $1/30$ of the input sites. Hence, we obtain the following recurrence on the time performance $T(n)$ of a parallel implementation of Algorithm 2 in the CREW PRAM model with $O(n/t(n))$ processors: $T(n) \leq \max_{1/3<\alpha<2/3}\{T(\alpha n)+T((1-\alpha-1/30)n)+O(t(n))\}$. Assume $T(n)$ to be of the form $O(n^\beta)$, where $\beta < 1$. Then, the worst case occurs when the difference between the argument of the first term and the argument of the second term on the right side of the recurrence is the smallest possible. Hence, we obtain the inequality $T(n) \leq 2T(29n/60)+O(t(n))$. This leads to the following estimation $T(n) = O(\sum_{i=0}^{\log_{60/29} n} 2^i t((29/60)^i n)$. Let δ be any positive constant greater than $\log_{60/29} 2$. Assume $t(n) = n^\delta$. By straightforward calculations, we obtain $T(n) = \sum_{i=0}^{\log_{60/29} n} (29/60)^{(\delta-\log_{60/29} 2)i} n^\delta$. Consequently, we have $T(n) = O(n^\delta)$ which yields the theorem thesis.

By Brent's principle [10], it is now sufficient to show that all the operations of Algorithm 2 but for the recursive calls can be implemented in time

$O(\log^k n)$ using $O(n/\log^k n)$ processors, where $k \in \{0,1\}$, in the CREW PRAM model.

- Step 1: The choice of the primary red sites involves only local computations and can be trivially done in constant time using a linear number of processors. The maximal subsequences of not yet colored consecutive sites can be found and then colored by parallel list ranking [10] in logarithmic time using $O(n/\log n)$ processors.

- Step 3: The prunning of the forest $V(B)$, identifying the twin leaves and choosing the ones with singleton subtrees in the prunned forest involve only local computations and can be done in constant time using a linear number of processors. By applying parallel list ranking to the contracted forest obtained by deleting its leaves we can identify the spines as maximal paths with nodes of degree two and divide them into groups with five leaves. It can be done in time $O(\log n)$ using $O(n/\log n)$ processors [10]. Finally, choosing a leaf with a trivial subtree or a leaf with a subtree confined to the group require only local computations and can be done for all groups in constant time using a linear number of processors.

- Step 4: The crimson sites have independent regions in $V(B \cup C)$ by Lemma 3.9. Therefore it is sufficient to construct $V(B \cup \{r\})$ in parallel for each r in C. The cost of the construction of $V(B \cup \{r\})$ is proportional to the size of the subtree T_l of the prunned forest T (see the discussion preceeding Lemma 3.9 and Step 3) contained in $V(B \cup \{r\})$. The crimson sites are chosen among the red sites in Step 3 (by applying Lemma 3.8). It follows from that all the subtrees T_l associated to the crimson sites r are of bounded size. Hence, this step can be implemented in constant time using a linear number of processors.

- Step 5: The garnet set can be clearly computed in $O(\log n)$ time using $O(n/\log n)$ processors in the CREW PRAM model, e.g., by applying a work-optimal logarithmic time sorting algorithm [4].

- Step 6: The diagrams $V(B \cup C)$ and $V(G)$ can be merged in $O(\log n)$ time using $O(n/\log n)$ CREW PRAM processors by applying the free-tree merging techniques due to Cole, Goodrich and Ó Dúnlaing [5] (see the proof of Theorem 7.2 in [5]).

□

5 Applications

Theorem 3.10 directly implies the result on convex polygons proved in [1].

To give applications of Theorem 3.3 we leave the abstract setting and turn to the family of *nice* metrics; see [11]. Roughly, a metric d is called nice iff it fulfills the following three conditions. i) Two points are close with respect to d if and only if they are close in the standard sense, ii) for any two points a, c there is a third point, b, between them such that $d(a,b) + d(b,c) = d(a,c)$ holds, and iii) bisectors of points are tractible. All semi-algebraic convex distance functions (see e. g. [11] for a definition) are nice; however, the class of nice metrics allows for phenomena that do not occur with convex distance functions: distance need not be invariant under translation, shortest paths need not be straight line segments, and the bisectors of three points can in fact cross twice.

If d is a metric then the *d-circle of radius r centered at p* is the set $K_d(r,p) = \{z; d(p,z) = r\}$. In the sequel we assume that the bisector $B_d(p,q) = \{z; d(p,z) = d(q,z)\}$ is always a curve (if not, one of its boundaries could be chosen, but we want to avoid these technical problems here.) Then Definitions 2.1 and 2.2 are fulfilled by the $B_d(p,q)$.

Lemma 5.1 *Let d be a nice metric, and let H be a simple curve homeomorphic to a line. Then the following assertions are equivalent.*

1. *For any finite set S of points on H is the Voronoi diagram $V(S)$ Hamiltonian with respect to H.*
2. *Each bisector $B_d(p,q)$, where p, q are points on H, crosses H only once.*
3. *Each circle $K_d(r,p)$, where p is a point on H, crosses H only twice.*
4. *For each point p on H, if we move away from p in either direction along H then the d-distance to p is strictly increasing.*

Proof: 1) $\Leftrightarrow$ 2) : If p lies on H then its region covers at least one interval of H. Thus, equivalence follows directly from Definition 3.2.
3) $\Leftrightarrow$ 4) : This is obvious.
2) $\Rightarrow$ 3) : Suppose that H runs through three points p, q, r on $K_d(r,v)$. Its center, v is a Voronoi vertex in $V(\{p,q,r\})$ lying on H. Since the simple curve H must visit each of the three sites it has to cross a Voronoi edge at a point z different from v. Thus, the corresponding bisector crosses H twice,

at v and at z.
4) $\Rightarrow$ 2) : As we walk from p to q along H, the distance to p is strictly increasing from zero to $d(p,q)$, while the distance from q is strictly decreasing from $d(p,q)$ to zero. Both functions are continuous. Thus, exactly one point of $B_d(p,q)$ will be crossed. Each point x on the unbounded piece of H that connects p to infinity and does not contain q is closer to p than to q. Namely, if one walks from p towards q then the distance to x increases. Therefore, this part of H does not cross the bisector. □

Definition 5.2 Let H be as in Lemma 5.1. Then H is called *Hamiltonian with respect to d.*

Corollary 5.3 *If d is a nice metric, and if n points are given in their order on a Hamiltonian curve with respect to d, then their Voronoi diagram based on d can be computed in time $O(n)$. Also, for any $\delta > \log_{60/29} 2$, the Voronoi diagram can be computed in time $O(n^\delta)$ using $O(n^{1-\delta})$ processors in the CREW PRAM model.*

Proof: Theorems 3.3, 4.1 and Lemma 5.1 □

For the Euclidean distance, examples of Hamiltonian curves are the graphs of functions $f(x) = y$ where f is convex or, more generally, monotone. This generalizes the results of [1] and [6], because one can simply split e. g. the contour of a convex polygon into four parts satisfying this condition, compute their diagrams separately, and merge them in linear time. Other examples are the graphs of smooth functions f satisfying $|f'(x)| \leq 1$, like the function $\sin x$.

6 Open Problems

In the proof of Theorem 3.3 it was very important that for each subset S' of S each region of $V(S')$ occurred on H *only once*, so we could apply recursion. There are cases where this property cannot be recursively maintained. An important example is the Voronoi diagram of the edges of a simple polygon P, inside P (also known as the skeleton, or medial axis structure, of P). The diagram is tree-shaped, and its regions are in a natural way ordered by the boundary. Currently, there is a deterministic $O(n \log n)$ algorithm by Yap [17] that allows to compute the diagram of an arbitrary set of line segments,

and a randomized $O(n \log^* n)$ algorithm by Devillers [7]. Whether or not a faster deterministic solution is possible remains open.

Also, the problem of dessigning faster parallel algorithms for Hamiltonian diagrams work optimal or nearly work is widely open.

References

[1] A. Aggarwal, L.J. Guibas, J. Saxe, and P.W. Shor. A Linear-Time Algorithm for Computing the Voronoi Diagram of a Convex Polygon. Discrete and Computational Geometry 2, 1987, Springer Verlag.

[2] F. Aurenhammer. Voronoi Diagrams—A Survey of a Fundamental Geometric Data Structure. ACM Computing Surveys, 23(3):345–405, 1991.

[3] C. Clarkson and P.W. Shor. Applications of Random Sampling in Computational Geometry II. Discrete and Computational Geometry, 4:387–421, 1989.

[4] R. Cole. *Parallel merge sort.* SIAM J. Computing, 17 (1988), pp. 770-785.

[5] R. Cole, M.T. Goodrich and C. Ó Dúnlaing. Merging Free Trees in Parallel for Efficient Voronoi Diagram Construction. Proc. 17th ICALP, LNCS 443, Springer Verlag, pp. 432-445.

[6] H. Djidjev and A. Lingas. On Computing the Voronoi Diagram for Restricted Planar Figures. To appear in Proc. WADS'91, LNCS, Springer Verlag.

[7] O. Devillers. Randomization Yields Simple $O(n \log^* n)$ Algorithms for Difficult $\Omega(n)$ Problems. International Journal of Computational Geometry & Applications 2(1), pp. 97–111, 1992.

[8] H. Edelsbrunner and R. Seidel. Voronoi Diagrams and Arrangements. Discrete and Computational Geometry, 1:25–44, 1986.

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

[10] R. M. Karp and V. Ramachandran, Parallel Algorithms for Shared-Memory Machines. Handbook of Theoretical Computer Science, Edited by J. van Leeuwen, Volume 1, Elsevier Science Publishers B.V., 1990.

[11] R. Klein. Concrete and Abstract Voronoi Diagrams. LNCS 400, Springer Verlag, 1989.

[12] R. Klein, K. Mehlhorn, and S. Meiser. Randomized Incremental Construction of Abstract Voronoi Diagrams. Technical Report MPI-I-93-105, Max-Planck-Institut Saarbrücken, 1993. To appear in Computational Geometry: Theory and Applications, 1993.

[13] S. Meiser. Zur Konstruktion abstrakter Voronoidiagramme. Ph.D. Thesis, Universität des Saarlandes, Saarbrücken, 1993.

[14] F.P. Preparata and M.I. Shamos. Computational Geometry: An Introduction. Texts and Monographs in Theoretical Computer Science, Springer Verlag, New York, 1985.

[15] M.I. Shamos and D. Hoey. Closest Point Problems. In Proc. of the 16th Annual IEEE Symposium on Foundations of Computer Science, pp. 151–162, 1975.

[16] C. Yap. Personal Communication.

[17] C. Yap. An $O(n \log n)$ Algorithm for the Voronoi Diagram of a Set of Simple Curve Segments. Discrete and Computational Geometry 2, pp. 365–393, 1987.