Dynamic additively weighted Voronoi diagrams in 2D[*]

Menelaos I. Karavelas[1] and Mariette Yvinec[1]

INRIA Sophia-Antipolis, Project PRISME,
2004 Route des Lucioles, BP 93,
06902 Sophia-Antipolis Cedex, France
{Menelaos.Karavelas, Mariette.Yvinec}@sophia.inria.fr

Abstract. In this paper we present a dynamic algorithm for the construction of the additively weighted Voronoi diagram of a set of weighted points in the plane. The novelty in our approach is that we use the dual of the additively weighted Voronoi diagram to represent it. This permits us to perform both insertions and deletions of sites easily. Given a set $\mathcal{B}$ of n sites, among which h sites have a non-empty cell, our algorithm constructs the additively weighted Voronoi diagram of $\mathcal{B}$ in $O(nT(h) + h \log h)$ expected time, where $T(k)$ is the time to locate the nearest neighbor of a query site within a set of k sites. Deletions can be performed for all sites whether or not their cell is empty. The space requirements for the presented algorithm is $O(n)$. Our algorithm is simple to implement and experimental results suggest an $O(n \log h)$ behavior.

1 Introduction

One of the most well studied structures in computational geometry is the Voronoi diagram for a set of sites. Applications include retraction motion planning, collision detection, computer graphics or even networking and communication networks. There have been various generalizations of the standard Euclidean Voronoi diagram, including generalizations to L_p metrics, convex distance functions, the power distance, which yields the power diagram, and others. The sites considered include points, convex polygons, line segments, circles and more general smooth convex objects.

In this paper we are interested in the *Additively Weighted Voronoi diagram* or, in short, AW-Voronoi diagram. We are given a set of points and a set of weights associated with them. Let $d(\cdot, \cdot)$ denote the Euclidean distance. We define the distance $\delta(p, B)$ between a point p on the Euclidean plane $\mathbb{E}^2$ and a weighted point $B = \{b, r\}$ as $\delta(p, B) = d(p, b) - r$. If the weights are positive, the additively weighted Voronoi diagram can be viewed geometrically as the Voronoi diagram for a set of circles, the centers of which are the points and the radii of which are the corresponding weights. Points outside a circle have positive distance with respect to the circle, whereas points inside a circle have negative distance. The

[*] Work partially supported by the IST Programme of the EU as a Shared-cost RTD (FET Open) Project under Contract No IST-2000-26473 (ECG - Effective Computational Geometry for Curves and Surfaces).

Voronoi diagram does not change if all the weights are translated by the same quantity. Hence, in the sequel we assume that all the weights are positive. In the same context we use the term *site* to denote interchangeably a weighted point or the corresponding circle. We also define the distance between two sites $B_1 = \{b_1, r_1\}$, $B_2 = \{b_2, r_2\}$ on the plane to be $\delta(B_1, B_2) = d(b_1, b_2) - r_1 - r_2$. Note that $\delta(B_1, B_2)$ is negative if the two sites (circles) intersect at two points or if one is inside the other. If we assign every point on the plane to its closest site, we get a subdivision of the plane into regions. The closures of these regions are called *Voronoi cells*. The one-dimensional connected sets of points that belong to exactly two Voronoi cells are called *Voronoi edges*, whereas points that belong to at least three Voronoi cells are called *Voronoi vertices*. The collection of cells, edges and vertices is called the *Voronoi diagram*. A *bisector* between two sites is the locus of points that are equidistant from both sites. Unlike the case of the Euclidean Voronoi diagram of points in an AW-Voronoi diagram the cell of a given site may be empty. Such a site is called *trivial*. We can actually fully characterize trivial sites: a site is trivial if it is fully contained inside another site.

The first algorithm for computing the AW-Voronoi diagram appeared in [1]. The running time of the algorithm is $O(nc^{\sqrt{\log n}})$, where c is a constant, and it works only in the case of disjoint sites. The same authors presented in [2] another algorithm for constructing the AW-Voronoi diagram, which runs in $O(n \log^2 n)$ time. This algorithm uses the divide-and-conquer paradigm and works again only for disjoint sites. A detailed description of the geometric properties of the AW-Voronoi diagram, as well as an algorithm that treats intersecting sites can be found in [3]. The algorithm runs in $O(n \log^2 n)$ time, and also uses the divide-and-conquer paradigm. A sweep-line algorithm is described in [4] for solving the same problem. The set of sites is first transformed to a set of points by means of a special transformation, and then a sweep-line method is applied to the point set. The sweep-line algorithm runs in time $O(n \log n)$. Aurenhammer [5] suggests a lifting map of the two-dimensional problem to three dimensions, and reduces the problem of computing the AW-Voronoi Voronoi diagram in 2D to computing the power diagram of a set of spheres in 3D. The algorithm runs in $O(n^2)$ time, but it is the first algorithm for constructing the AW-Voronoi diagram that generalizes to dimension $d \geq 3$. If we do not have trivial sites, every pair of sites has a bisector. In this case, the AW-Voronoi diagram is a concrete type of an *Abstract Voronoi diagram* [6], for which optimal divide-and-conquer $O(n \log n)$ algorithms exist. Incremental algorithms that run in $O(n \log n)$ expected time also exist for abstract Voronoi diagrams (see [7, 8]). The algorithm in [8] allows the insertion of sites with empty Voronoi cell. However, it does not allow for deletions and the data structures used are a bit involved. More specifically, the Voronoi diagram itself is represented as a planar map and a history graph is used to find the conflicts of the new site with the existing Voronoi diagram. Finally, an off-line algorithm that constructs the Delaunay triangulation of the centers of the sites and then performs edge-flips in order to restore the AW-Delaunay graph is presented in [9]. Again this algorithm does not allow the deletion of sites, and moreover it does not handle the case of trivial sites.

In this paper we present a fully dynamic algorithm for the construction of the AW-Voronoi diagram. Our algorithm resembles the algorithm in [8], but, it also has several differences. Firstly, we do not represent the AW-Voronoi diagram, but rather its dual. The dual of the AW-Voronoi diagram is a planar graph of linear size [3], which we call the *Additively Weighted Delaunay graph* or AW-Delaunay graph, for short. Moreover, under the non-degeneracy assumption that there are no points in the plane equidistant to more than 3 sites, the AW-Delaunay graph has triangular faces. Our algorithm requires no assumption about degeneracies. It implicitly uses a perturbation scheme which simulates non-degeneracies and yields an AW-Delaunay graph with triangular faces. Hence, representing the AW-Voronoi diagram can be done in a much simpler way compared to [8]. In [8] the insertion is done in two stages. First the history graph is used to find the conflicts of the new site with the existing AW-Voronoi diagram. Then both the planar map representation of the AW-Voronoi diagram and the history graph are updated, in order to incorporate the Voronoi cell of the new site. In our algorithm the insertion of the new site is done in three stages. The first stage finds the nearest neighbor of the new site in the existing AW-Voronoi diagram. Using the nearest neighbor, the second stage determines if the new site is trivial. Starting from the nearest neighbor of the new site, the third stage finds all the Voronoi edges in conflict with the new site and then reconstructs the AW-Voronoi diagram. This is done using the dual graph. Another novelty of our algorithm is that it permits the deletion of sites, which is not the case in [8].

The remainder of the paper is structured as follows. In Section 2 we define the AW-Voronoi and its dual. We review some of the known properties of the AW-Voronoi diagram and provide definitions used in the remainder of the paper. In Section 3 we show how to insert a new site, and discuss how we deal with trivial sites. In Section 4 we describe how to delete sites. In Section 5 we briefly discuss the predicates involved in our algorithm and present experimental results. A more detailed analysis of the predicates and how to compute them can be found in [10]. Section 6 is devoted to conclusions and directions for further research. Due to space limitations the proofs of the lemmas and theorems are omitted. this version of the paper. The interested reader can find these proofs in [11].

2 Preliminaries

Let $\mathcal{B}$ be a set of sites B_j, with centers b_j and radii r_j. For each $j \neq i$, let $H_{ij} = \{y \in \mathbb{E}^2 : \delta(y, B_i) \leq \delta(y, B_j)\}$. Then the (closed) Voronoi cell V_i of B_i is defined to be $V_i = \cap_{i \neq j} H_{ij}$. The connected set of points that belong to exactly two Voronoi cells are called *Voronoi edges*, whereas points that belong to at least three Voronoi cells are called *Voronoi vertices*. The *AW-Voronoi diagram* $\mathcal{V}(\mathcal{B})$ of $\mathcal{B}$ is defined as the collection of the Voronoi cells, edges and vertices. The *Voronoi skeleton* $\mathcal{V}_1(\mathcal{B})$ of $\mathcal{B}$ is defined as the union of the Voronoi edges and Voronoi vertices of $\mathcal{V}(\mathcal{B})$. The AW-Voronoi diagram is a subdivision of the plane

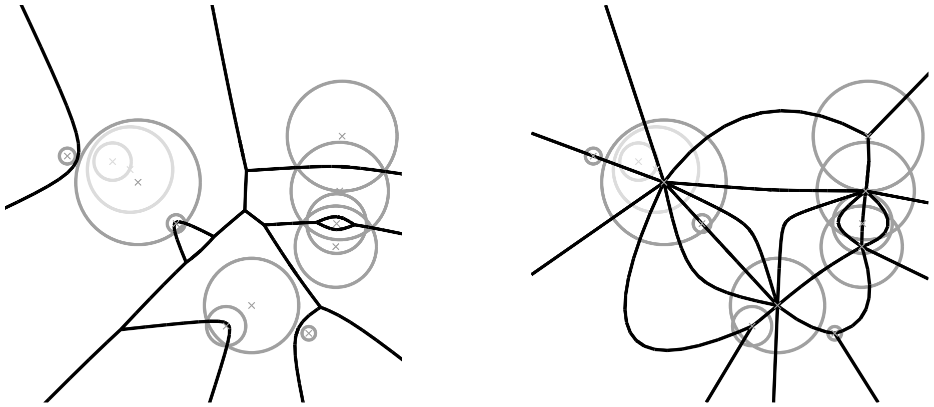

Fig. 1. Left: the AW-Voronoi diagram for a set of 12 sites, among which 2 sites are trivial. Non-trivial sites are shown in gray. Trivial sites are shown in light gray. The Voronoi skeleton is shown in black. Right: a planar embedding of the AW-Delaunay graph of the same set of sites. The edges of the AW-Delaunay graph are shown in black.

[3, Property 1]. The Voronoi edges are straight or hyperbolic arcs and each cell is star-shaped with respect to the center of the corresponding site [3, Properties 3 and 4]). In the case of the usual Euclidean Voronoi diagram for a set of points, every point has a non-empty Voronoi cell. In AW-Voronoi diagrams there may exist sites with empty Voronoi cells. In particular, the Voronoi cell V_i of a site B_i is empty if and only if B_i is contained in another site B_j [3, Property 2]. A site whose Voronoi cell has empty interior is called *trivial*, whereas a site whose Voronoi cell has non-empty interior is called *non-trivial* (see Fig. 1(left)).

We call AW-Delaunay graph, and denote by $\mathcal{D}(\mathcal{B})$, the dual graph of the AW-Voronoi diagram $\mathcal{V}(\mathcal{B})$. There is a vertex in $\mathcal{D}(\mathcal{B})$ for each non-trivial site B_i in $\mathcal{B}$. Let B_i and B_j be two sites whose Voronoi cells V_i and V_j are adjacent. We denote by α_{ij}^{kl} the Voronoi edge in $V_i \cap V_j$ whose endpoints are the Voronoi vertices equidistant to B_i, B_j, B_k and B_i, B_j, B_l, respectively. There exists an edge e_{ij}^{kl} in $\mathcal{D}(\mathcal{B})$ connecting B_i and B_j for each edge α_{ij}^{kl} of $\mathcal{V}(\mathcal{B})$ in $V_i \cap V_j$. The fact that we have a planar embedding of linear size for the AW-Delaunay graph [3, Property 7] immediately implies that the size of the AW-Voronoi diagram is $O(n)$. The Voronoi skeleton may consist of more than one connected component [3, Property 9], whereas the dual graph is always connected. If we do not have any degeneracies, the AW-Delaunay graph has the property that all but its outer face have exactly three edges. However, it may contain vertices of degree 2, i.e., we have triangular faces with two edges in common. If the Voronoi skeleton consists of more than one connected component the AW-Delaunay graph may also have vertices of degree 1, which are the dual of Voronoi cells with no vertices (e.g., the Voronoi cell at the top left corner of Fig. 1(left)). To simplify the representation of the AW-Delaunay graph we add a fictitious site called the site at infinity. This amounts to adding a Voronoi vertex on each unbounded edge of $\mathcal{V}_1(\mathcal{B})$. These additional vertices are then connected through Voronoi edges forming the boundary of the infinite site cell. In this compactified version, the Voronoi skeleton consists of only one connected component, and the previously non-connected components are now connected through the edges of the Voronoi cell of the site at infinity. The compactified AW-Delaunay graph corresponds to the original AW-Delaunay graph plus edges connecting the sites on the convex hull of $\mathcal{B}$ with the site at infinity. In the absence of degeneracies, all faces of the compactified AW-Delaunay graph have exactly three edges, but this graph may still have vertices of degree 2. From now on when we refer to the AW-Voronoi diagram or the AW-Delaunay graph, we refer to their compactified versions (see Fig. 1(right)). Degenerate cases arise when there are points equidistant to more than three sites. Then, the AW-Delaunay graph has faces with more than three edges. This is entirely analogous to the situation for the usual Delaunay diagram for a set of points with subsets of more than three cocircular points. In such a case, a graph with triangular faces can be obtained from the AW-Delaunay graph through an arbitrary triangulation of the faces with more than three edges. Our algorithm uses an implicit perturbation scheme and produces in fact such a triangulated AW-Delaunay graph.

Let B_i and B_j be two sites such that no one is contained inside the other. A circle tangent to B_i and B_j that neither contains any of them nor is contained in any of them is called an *exterior bitangent Voronoi circle*. A circle tangent to B_i and B_j that lies in $B_i \cap B_j$ is an *interior bitangent Voronoi circle*. Similarly, a circle tangent to three sites B_i, B_j and B_k is an *exterior tritangent Voronoi circle* if it neither contains any of B_i, B_j and B_k nor is contained in any of them. A circle tangent to B_i, B_j and B_k is called an *interior tritangent Voronoi circle* if it is included in $B_i \cap B_j \cap B_k$. A triple of sites B_i, B_j and B_k can have up to two tritangent Voronoi circles, either exterior or interior. This is equivalent to stating that the AW-Voronoi diagram of three sites can have up to two Voronoi vertices [3, Property 5]. Let π_{ij} denote the bisector of the sites B_i and B_j. As we already mentioned π_{ij} can be a line or a hyperbola. We define the orientation of π_{ij} to be such that b_i is always to the left of π_{ij}. The orientation of π_{ij} defines an ordering on the points of π_{ij}, which we denote by $\prec_{ij}$. Let o_{ij} be the intersection of π_{ij} with the segment b_ib_j. We can parameterize π_{ij} as follows: if $o_{ij} \prec_{ij} p$ then $\zeta_{ij}(p) = \delta(p, B_i) - \delta(o_{ij}, B_i)$; otherwise $\zeta_{ij}(p) = -(\delta(p, B_i) - \delta(o_{ij}, B_i))$. The function $\zeta_{ij}(\cdot)$ is a 1–1 and onto mapping from π_{ij} to $\mathbb{R}$. The *shadow region* $S_{ij}(B)$ of a site B with respect to the bisector π_{ij} of B_i and B_j is the locus of points c on π_{ij} such that $\delta(B, C_{ij}(c)) < 0$, where $C_{ij}(c)$ is the bitangent Voronoi circle of B_i and B_j centered at c. Let $\tilde{S}_{ij}(B)$ denote the set of parameter values $\zeta_{ij}(c)$, where $c \in S_{ij}(B)$. It is easy to verify that $\tilde{S}_{ij}(B)$ can be of the form $\emptyset$, $(-\infty, \infty)$, $(-\infty, a)$, (b, ∞), (a, b) and $(-\infty, a) \cup (b, \infty)$, where $a, b \in \mathbb{R}$.

Let α_{ij}^{kl} be an edge of $\mathcal{V}(\mathcal{B})$ on the bisector π_{ij}. Let C_{ijk} and C_{ijl} be the tritangent Voronoi circles associated with the endpoints of α_{ij}^{kl}. We denote by c_{ijk} (resp. c_{ijl}) the center of C_{ijk} (resp. C_{ijl}). Under the mapping $\zeta_{ij}(\cdot)$, α_{ij}^{kl} maps to the interval $\tilde{\alpha}_{ij}^{kl} = [\xi_{ijl}, \xi_{ijk}] \subset \mathbb{R}$. We define the *conflict region* $R_{ij}^{kl}(B)$ of B with respect to the edge α_{ij}^{kl} to be the intersection $R_{ij}^{kl}(B) = \alpha_{ij}^{kl} \cap S_{ij}(B)$. We say that

B *is in conflict* with α_{ij}^{kl} if $R_{ij}^{kl}(B) \neq \emptyset$. Under the mapping by $\zeta_{ij}(\cdot)$, the conflict region $R_{ij}^{kl}(B)$ maps to the intersection $\tilde{R}_{ij}^{kl}(B) = \tilde{\alpha}_{ij}^{kl} \cap \tilde{S}_{ij}(B)$. $\tilde{R}_{ij}^{kl}(B)$ can be one of the following types : (1) $\tilde{R}_{ij}^{kl}(B) = \emptyset$. We say that B *is not in conflict* with α_{ij}^{kl}. (2) $\tilde{R}_{ij}^{kl}(B)$ consists of two disjoint intervals, including respectively ξ_{ijk} and ξ_{ijl}. We say that B *is in conflict with both vertices of* α_{ij}^{kl}. (3) $\tilde{R}_{ij}^{kl}(B)$ consists of a single connected interval. We further distinguish between the following cases : (a) $\tilde{\alpha}_{ij}^{kl} = \tilde{R}_{ij}^{kl}(B)$. We say that B *is in conflict with the entire edge* α_{ij}^{kl}. (b) $\tilde{R}_{ij}^{kl}(B)$ contains either ξ_{ijk} or ξ_{ijl}, but not both. We say that B *is in conflict with one vertex of* α_{ij}^{kl}. (c) $\tilde{R}_{ij}^{kl}(B) \neq \emptyset$ but contains neither ξ_{ijk} nor ξ_{ijl}. We say that B *is in conflict with the interior of* α_{ij}^{kl}. Finally we define the *conflict region* $R_{\mathcal{B}}(B)$ of B with respect to $\mathcal{B}$ as the union $R_{\mathcal{B}}(B) = \bigcup_{\alpha_{ij}^{kl} \in \mathcal{V}(\mathcal{B})} R_{ij}^{kl}(B)$. It is easy to verify that $R_{\mathcal{B}}(B) = V_{\mathcal{B} \cup \{B\}}(B) \cap \mathcal{V}_1(\mathcal{B})$, where $V_{\mathcal{B} \cup \{B\}}(B)$ denotes the Voronoi cell of B in $\mathcal{V}(\mathcal{B} \cup \{B\})$.

3 Inserting a site incrementally

In this section we present the incremental algorithm. Let again $\mathcal{B}$ be our set of n sites and let us assume that we have already constructed the AW-Voronoi diagram for a subset $\mathcal{B}_m$ of $\mathcal{B}$. Here m denotes the number of sites in $\mathcal{B}_m$. We now want to insert a site $B \notin \mathcal{B}_m$. The insertion is done in the following steps : (i) locate the nearest neighbor $NN(B)$ of B in $\mathcal{B}_m$; (ii) test if B is trivial; (iii) find the conflict region of B and repair the AW-Delaunay graph. We postpone the discussion on the location of the nearest neighbor until the end of this section.

The first test we have to do is to determine whether B is trivial or not. This can easily be done once we know the nearest neighbor $NN(B)$ of B, since B is trivial if and only if $B \subset NN(B)$ [11, Lemma 1]. Let $R_m(B)$ be the conflict region of B with respect to $\mathcal{B}_m$. Let $\partial R_m(B)$ denote the boundary of $R_m(B)$. $R_m(B)$ is a subset of $\mathcal{V}_1(\mathcal{B}_m)$ and $\partial R_m(B)$ is a set of points on edges of $\mathcal{V}_1(\mathcal{B}_m)$. Points in $\partial R_m(B)$ are the vertices of the Voronoi cell V_B of B in $\mathcal{V}(\mathcal{B}_{m+1})$, where $\mathcal{B}_{m+1} = \mathcal{B}_m \cup \{B\}$. It has been shown in [8, Lemma 1] that $R_m(B)$ is connected. Thus, the aim is to discover the boundary $\partial R_m(B)$ of $R_m(B)$, since then we can repair the AW-Voronoi diagram in exactly the same way as in [8]. The idea is to perform a *depth first search* (DFS) on $\mathcal{V}_1(\mathcal{B}_m)$ to discover $R_m(B)$ and $\partial R_m(B)$, starting from a point on the skeleton that is known to be in conflict with B. Let L denote the boundary of the currently discovered portion of $R_m(B)$. Initially $L = \emptyset$. We are going to represent points in L by the Voronoi edges that contain them. We want the points of $\partial R_m(B)$ to appear in L in the order that they appear on the boundary of the Voronoi region V_B of B in $\mathcal{V}(\mathcal{B}_{m+1})$. Without loss of generality we can choose this order to be the counter-clockwise ordering of the vertices on the boundary of V_B.

As we mentioned in the previous paragraph, we need to find a first point on the Voronoi skeleton $\mathcal{V}_1(\mathcal{B}_m)$, that is in conflict with B. This point is going to serve as the starting point for the DFS. It can be shown that if B is a non-trivial site, then B has to be in conflict with at least one of the edges of the Voronoi cell

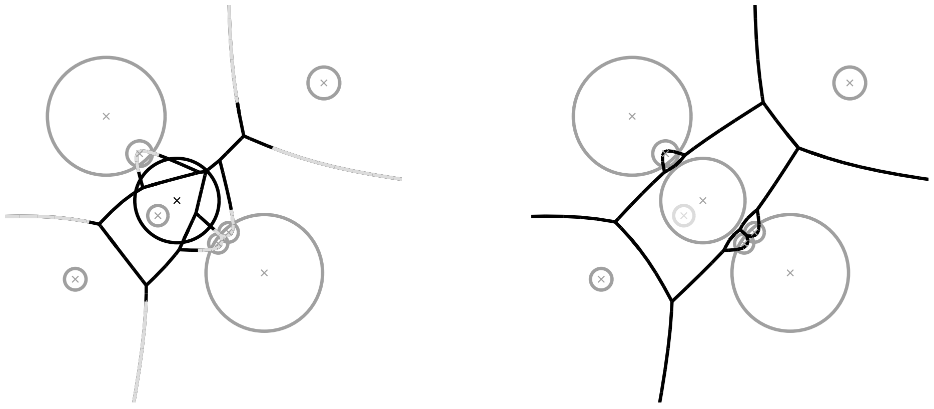

Fig. 2. Left: The AW-Voronoi diagram for a set of sites (gray) and the conflict region (black) of a new site (also black). The portion of the Voronoi skeleton that does not belong to the conflict region of the new site is shown in light gray. Right: The AW-Voronoi diagram after the insertion of the new site. Non-trivial sites, including the new site, are shown in gray. The site in light gray is inside the new site and has become trivial. The Voronoi skeleton is shown in black.

$V_{NN(B)}$ of $NN(B)$ in $\mathcal{V}(\mathcal{B}_m)$ [11, Lemma 2]. Hence, we can simply walk on the boundary of $V_{NN(B)}$, until we find a Voronoi edge in conflict with B. Let α be the first edge, of the boundary of $V_{NN(B)}$ that we found to be in conflict with B. If B is in conflict with the interior of α, we have discovered the entire conflict region $R_m(B)$. In this case L consists of two copies of α with different orientations. Otherwise, B has to be in conflict with at least one of the two Voronoi vertices of α. In this case we set L to be the edges adjacent to that Voronoi vertex in counter-clockwise order. The DFS will then recursively visit all vertices in conflict with B. Suppose that we have arrived at a Voronoi vertex v (which is a node on the Voronoi skeleton). Firstly, we mark it. Then we look at all the Voronoi edges α adjacent to it. Let v' be the Voronoi vertex of α that is different from v. We consider the following two cases : (1) $\underline{v' \text{ has not been marked}}$. If B is in conflict with the entire edge α, then we replace α in L by the remaining Voronoi edges adjacent to v', in counter-clockwise order. We then continue recursively on v'. If B is not in conflict with the entire edge α, we have reached an edge α containing a point of $\partial R_m(B)$. The list L remains unchanged and the DFS backtracks. (2) $\underline{v' \text{ has already been marked}}$. If B is in conflict with the entire edge α, have found a cycle in $R_m(B)$, or equivalently, B contains a site in $\mathcal{B}_m$, which will become trivial. Since α currently belongs to L, but does not contain any points of $\partial R_m(B)$, we remove it from L. The DFS then backtracks. If B is not in conflict with the entire edge α, then B is in conflict with both vertices of α. Hence α contains two points of $\partial R_m(B)$ in its interior. The list L remains unchanged and the DFS backtracks. Note that in this case α appears twice in L, once per point in $\partial R_m(B)$ that it contains. Fig. 2(top left) shows an example of a conflict region which triggers all the possible cases of the above search algorithm. In our case, the AW-Voronoi diagram is represented through its dual AW-Delaunay graph. It is thus convenient to restate the algorithm for finding the boundary $\partial R_m(B)$ of the conflict region $R_m(B)$ in terms of the AW-Delaunay graph. This is done in the long version of this paper (cf. [11]).

In case of degeneracies, the algorithm uses a perturbation scheme described by the following lazy strategy. Any new site which is found tangent to a tritangent Voronoi circle is considered as not being in conflict with the corresponding Voronoi vertex. Then any Voronoi vertex remains a degree 3 vertex and the dual AW-Delaunay graph is always triangular. This graph, however, is not canonical, but depends on the insertion order of the sites.

During the insertion procedure trivial sites can appear in two possible ways. Either the new site B to be inserted is trivial, or B contains existing sites, which after the insertion of B will become trivial. When deletion of sites is allowed, B may contain other sites which will become non-trivial if B is deleted. For this reason we need to keep track of trivial sites. Since a site is trivial if and only if it is contained inside some other site, there exists a natural parent-child relationship between trivial and non-trivial sites. In particular, we can associate every trivial site to a non-trivial site that contains it. If a trivial site is contained in more than one non-trivial sites, we can choose the parent of the trivial site arbitrarily. A natural choice for storing trivial sites is to maintain, for every non-trivial site, a list containing all trivial sites that have the non-trivial site as their parent. Let $\mathcal{B}_m^+$ be the subset of non-trivial sites of $\mathcal{B}_m$, and let $\mathcal{B}_m^- = \mathcal{B}_m \setminus \mathcal{B}_m^+$. For some $B' \in \mathcal{B}_m^+$, we define $L_{tr}(B')$ to be the list of trivial sites in $\mathcal{B}_m^-$ that have B' as their parent. We note by $\mathcal{L}_m$ the set of all lists $L_{tr}(B')$ for $B' \in \mathcal{B}_m^+$, and correspondingly $\mathcal{L}_{m+1}$ the set of all $L_{tr}(B')$ for $B' \in \mathcal{B}_{m+1}^+$. When a new site B is inserted and B is found to be trivial, we simply add B to $L_{tr}(NN(B))$. If B is non-trivial, let $\mathcal{B}_m^-(B)$ be the set of sites in $\mathcal{B}_m^+$ that are contained in B. Since after the insertion of B all sites in $\mathcal{B}_m^-(B)$ become trivial, we add every $B'' \in \mathcal{B}_m^-(B)$ to $L_{tr}(B)$. Moreover, for every $B'' \in \mathcal{B}_m^-(B)$ we move all sites in $L_{tr}(B'')$ to $L_{tr}(B)$. The following theorem subsumes the run time analysis of our algorithm. A detailed proof can be found in [11].

Theorem 1. *Let $\mathcal{B}$ be a set of n sites among which h are non-trivial. We can construct the AW-Voronoi diagram incrementally in $O(nT(h) + h \log h)$ expected time, where $T(k)$ is the time to locate the nearest neighbor of a query site within a set of k sites.*

We now turn our discussion on the location of the nearest neighbor. The nearest neighbor location of B in fact reduces to the location of the center b of B in $\mathcal{V}(\mathcal{B}_m)$. We can do that as follows. Select a site $B' \in \mathcal{B}_m$ at random. Look at all the neighbors of B' in the AW-Delaunay graph. If there exists a B'' such that $\delta(B, B'') < \delta(B, B')$, then B' cannot be the nearest neighbor of B. In this case we replace B' by B'' and restart our procedure. If none of the neighbors of B' is closer to B than B', then $NN(B) = B'$. The time to find the nearest neighbor using the above procedure is trivially $O(h)$, where h is the number of non-trivial sites in $\mathcal{B}$. However, we can speed-up the nearest-neighbor location by maintaining a hierarchy of AW-Delaunay graphs as is done in [12] for the Delaunay triangulation for points. The details can be found in [11]. The randomized time analysis for the location and insertion of a point in the Delaunay hierarchy has been given in [12]. Unfortunately, this analysis does not generalize to the AW-Delaunay hierarchy. Our experimental results, however, show that we do get a speed-up and that in practice the nearest-neighbor location is done in time $O(\log h)$, which gives a total running time of $O(n \log h)$ (see Section 5).

4 Deleting a site

Suppose that we have been given a set $\mathcal{B}$ of sites for which we have already constructed the AW-Voronoi diagram $\mathcal{V}(\mathcal{B})$. Let also $B \in \mathcal{B}$ be a site that we want to delete from $\mathcal{V}(\mathcal{B})$. In this section we describe how to perform the deletion.

Suppose that B is non-trivial. Let $\mathcal{B}_\gamma$ be the set of neighbors of V_B in $\mathcal{D}(\mathcal{B})$. Let also $L_{tr}^+(B)$ be the set of sites in $L_{tr}(B)$ that become non-trivial after the deletion of B. Finally, let $L_{tr}^-(B) = L_{tr}(B) \setminus L_{tr}^+(B)$, $\mathcal{B}_s = \mathcal{B}_\gamma \cup L_{tr}(B)$ and $\mathcal{B}_s^+ = \mathcal{B}_\gamma \cup L_{tr}^+(B)$. The main observation is that the second nearest neighbor of each point in V_B is one of the sites in $\mathcal{B}_s^+$. Moreover, every site in $L_{tr}^-(B)$ is inside one of the sites in $\mathcal{B}_s^+$ [11, Lemma 6]. Consequently, the AW-Voronoi diagram after the deletion of $\mathcal{B}$ can be found by constructing the AW-Voronoi diagram of $\mathcal{B}_\gamma \cup L_{tr}(B)$. More precisely, if b is a degree 3 vertex in $\mathcal{D}(\mathcal{B})$ and $|L_{tr}(B)| = 0$, we simply remove from $\mathcal{D}(\mathcal{B})$ the vertex corresponding to B as well as all its incident edges. If b is a degree 2 vertex in $\mathcal{D}(\mathcal{B})$ and $|L_{tr}(B)| = 0$, we again remove from $\mathcal{D}(\mathcal{B})$ the vertex v_B corresponding to B as well as all its incident edges. In addition, we collapse the edges e and e', where e and e' are the two edges of the star of v_B that are not incident to v_B. If the degree of b in $\mathcal{D}(\mathcal{B})$ is at least 4 or if $|L_{tr}(B)| > 0$, we construct $\mathcal{V}(\mathcal{B}_s)$ and then we find the nearest neighbor of B in $\mathcal{B}_s$. Once the nearest neighbor has been found we compute the conflict region of B in $\mathcal{B}_s$ by means of the procedure described in Section 3. Let $\partial^* R_s(B)$ be the representation, by means of the dual edges, of the conflict region of B in $\mathcal{B}_s$. The edges in $\partial^* R_s(B)$ are the dual of the AW-Voronoi edges in $\partial R_s(B)$. The triangles inside $\partial^* R_s(B)$ are the triangles that must appear in the interior of the boundary of the star of B when B is deleted from $\mathcal{D}(\mathcal{B})$. Therefore we can use these triangles to construct $\mathcal{D}(\mathcal{B} \setminus \{B\})$, or equivalently $\mathcal{V}(\mathcal{B} \setminus \{B\})$. Finally, all lists in $\mathcal{L}(\mathcal{B}_s^+)$ must be merged with their corresponding lists in $\mathcal{L}(\mathcal{B} \setminus \{B\})$.

Suppose that B is trivial. In this case we have to find the non-trivial site B' such that $B \in L_{tr}(B')$ and then delete B from $L_{tr}(B')$. Since $B \subset NN(B)$, B must be in the list of some B', which is in the same connected component of the union of sites as $NN(B)$. It has been shown that the subgraph $\mathcal{K}(\mathcal{B})$ of $\mathcal{D}(\mathcal{B})$ that consists of all edges of $\mathcal{D}(\mathcal{B})$ connecting intersecting sites, is a spanning subgraph of the connectivity graph of the set of sites [13, Chapter 5]. The deletion of a trivial site can, thus, be done as follows : (i) find the nearest neighbor $NN(B)$ of B; (ii) walk on the connected component $\mathcal{C}$ of $NN(B)$ in the graph $\mathcal{K}(\mathcal{B})$ and for every site $B' \in \mathcal{C}$ that contains B, test if $B \in L_{tr}(B')$; (iii) once the site B', such that $B \in L_{tr}(B')$, is found, delete B from $L_{tr}(B')$. The following theorem discusses the cost of deleting a site. A detailed analysis can be found in [11].

Theorem 2. *Let $\mathcal{B}$ be a set of n sites, among which h are non-trivial. Let $B \in \mathcal{B}$, and let $L_{tr}(B)$ be the list of trivial sites whose parent is B. If B is non-trivial, it can be deleted from $\mathcal{V}(\mathcal{B})$ in expected time $O((d+t)T(d+t')+(d+t')\log(d+t'))$, where d is the degree of B in $\mathcal{D}(\mathcal{B})$, t is the cardinality of $L_{tr}(B)$ and t' is the number of sites in $L_{tr}(B)$ that become non-trivial after the deletion of B. If B is trivial it can be deleted from $\mathcal{L}(\mathcal{B})$ in worst case time $O(n)$.*

5 Predicates and implementation

For the purposes of computing the algebraic degree of the predicates used in our algorithm, we assume that each site is given by its center and its radius. The predicates that we use are the following :

1. Given two sites B_1 and B_2, and a query site B, determine if B is closer to B_1 or B_2. This is equivalent to comparing the distances $\delta(b, B_1)$ and $\delta(b, B_2)$. This predicate is used during the nearest neighbor location phase and it is of algebraic degree 4 in the input quantities.
2. Given a site B_1 and a query site B, determine if $B \subset B_1$. This is equivalent to the expression $\delta(B, B_1) < -2r$, where r is the radius of B. This predicate is used during the insertion procedure in order to determine whether the query site is trivial. The algebraic degree of the predicate is 2.
3. Given two sites B_1 and B_2 determine if they intersect. This predicate is used during the deletion of a trivial site, and its algebraic degree is 2.
4. Given two sites B_1 and B_2 and a tritangent Voronoi circle C_{345} determine the result of the orientation test $\texttt{CCW}(b_1, b_2, c_{345})$, where b_1, b_2 and c_{345} are the centers of B_1, B_2 and C_{345}, respectively. This predicate is used in order to find the first conflict of a new site B given its nearest neighbor $NN(B)$. The evaluation of this predicate is discussed in [10], where is it also shown that its algebraic degree is 14.
5. Given a Voronoi edge α and a query site B, determine the type of the conflict region of B with α. This predicate is used in order to discover the conflict region of B with respect to the existing AW-Voronoi diagram. A method for evaluating this predicate is presented in [10]. The corresponding algebraic degree is shown to be 16 in the input quantities, using techniques from Sturm sequences theory.

We have implemented two versions of our algorithm, which differ only on how the nearest neighbor location is done. The first one does the nearest neighbor location using the simple procedure described in Section 3. The second implementation maintains a hierarchy of AW-Delaunay graphs. The predicates are evaluated exactly and have been implemented using two scenarios. The two scenarios are adapted, respectively, to number types that support the operations

n	h	h/n	T_1	T_2	$T_1/(n \log h)$	$T_2/(n \log h)$
10 000	10 000	1.00	4.75	3.59	1.18×10^{-4}	0.90×10^{-4}
10 000	7 973	0.80	4.46	3.65	1.14×10^{-4}	0.94×10^{-4}
10 000	5 017	0.50	3.64	3.02	0.98×10^{-4}	0.82×10^{-4}
100 000	99 995	1.00	85.17	38.42	1.70×10^{-4}	0.77×10^{-4}
100 000	79 861	0.80	83.37	38.52	1.70×10^{-4}	0.79×10^{-4}
100 000	49 614	0.50	67.15	32.19	1.43×10^{-4}	0.68×10^{-4}
1 000 000	999 351	1.00	$>$ 36 min	425.38	–	0.71×10^{-4}
1 000 000	800 290	0.80	2 130.49	445.58	3.61×10^{-4}	0.75×10^{-4}
1 000 000	497 866	0.50	1 715.94	386.47	3.01×10^{-4}	0.68×10^{-4}

Table 1. The running times of the two algorithms as a function of the size n of the input set and the number of non-trivial sites h. T_1 indicates the time for the algorithm with one level of the AW-Voronoi diagram and T_2 indicates the running time for an hierarchy of AW-Voronoi diagrams. Unless otherwise indicated, both T_1 and T_2 are given in seconds. The experiments were performed on a Pentium-III 1GHz running Linux.

$\{+, -, \times, /, \sqrt{\ }\}$ and $\{+, -, \times\}$ exactly. Both algorithms were implemented in C++, following the design of the library CGAL [14]. Our C++ code also supports CGAL's dynamic filtering [15], which is also used in our experiments. Finally, our experimental results were produced using the implementations of the predicates that do not use square roots for their evaluation. The two algorithms were tested on random circle sets of size $n \in \{10^4, 10^5, 10^6\}$ (see Table 1). The centers were uniformly distributed in the square $[-M, M] \times [-M, M]$, where $M = 10^6$. The radii of the circles were uniformly distributed in the interval $[0, R]$, were R was chosen appropriately so as to achieve different ratios h/n. In particular, we chose R so that the ratio h/n is approximately equal to the values in the set $\{1.00, 0.80, 0.50\}$. The last column of Table 1 suggests that our algorithm runs in time $O(n \log h)$ if we use the AW-Delaunay hierarchy. The algorithm with one level of the AW-Delaunay graph performs well for small inputs, but it is not a good choice for data sets where n is large and $h = \Theta(n)$.

6 Conclusion

This paper proposes a dynamic algorithm to compute the additively weighted Voronoi diagram for a set of weighted points in the plane. The algorithm represents the AW-Voronoi diagram through its dual graph, the AW-Delaunay graph and allows the user to perform dynamically insertions and deletions of sites. Given a set of n sites, among which h have non-empty cell, our algorithm constructs the AW-Voronoi diagram in expected time $O(nT(h) + h \log h)$, where $T(k)$ is the time to locate the nearest neighbor of a site within a set of k sites with non-empty Voronoi cell. Two methods are proposed to locate the nearest neighbor of a given site. The first one uses no additional data structure, performs a simple walk in the AW-Delaunay graph and locates the nearest neighbor in

$O(h)$ worst case time. The second method maintains a hierarchy of AW-Delaunay graphs, analog to the Delaunay hierarchy, and uses this hierarchy to perform the nearest neighbor location. Although the analysis of the Delaunay hierarchy does not extend to the case of the AW-Delaunay hierarchy, experimental results suggest that such a hierarchy allows to answer a nearest neighbor query in $O(\log h)$ time. Our algorithm performs deletions of non-trivial sites in almost optimal time. However, deletions of trivial sites are not done very efficiently and this point should be improved in further studies.

Further works also include generalization of our method to more general classes of objects, such as convex objects. More generally, one can think of characterizing classes of abstract Voronoi diagrams that can be computed using the method proposed here, i.e., without using a history or conflict graph. Another natural direction of future research is the generalization of the presented algorithm for the construction of AW-Voronoi diagrams in higher dimensions.

References

1. Drysdale, III, R.L., Lee, D.T.: Generalized Voronoi diagrams in the plane. In: Proc. 16th Allerton Conf. Commun. Control Comput. (1978) 833–842
2. Lee, D.T., Drysdale, III, R.L.: Generalization of Voronoi diagrams in the plane. SIAM J. Comput. **10** (1981) 73–87
3. Sharir, M.: Intersection and closest-pair problems for a set of planar discs. SIAM J. Comput. **14** (1985) 448–468
4. Fortune, S.: A sweepline algorithm for Voronoi diagrams. In: Proc. 2nd Annu. ACM Sympos. Comput. Geom. (1986) 313–322
5. Aurenhammer, F.: Power diagrams: properties, algorithms and applications. SIAM J. Comput. **16** (1987) 78–96
6. Klein, R.: Concrete and Abstract Voronoi Diagrams. Volume 400 of Lecture Notes Comput. Sci. Springer-Verlag (1989)
7. Mehlhorn, K., Meiser, S., Ó'Dúnlaing, C.: On the construction of abstract Voronoi diagrams. Discrete Comput. Geom. **6** (1991) 211–224
8. Klein, R., Mehlhorn, K., Meiser, S.: Randomized incremental construction of abstract Voronoi diagrams. Comput. Geom. Theory Appl. **3** (1993) 157–184
9. Kim, D.S., Kim, D., Sugihara, K.: Voronoi diagram of a circle set constructed from Voronoi diagram of a point set. In Lee, D.T., Teng, S.H., eds.: Proc. 11th Inter. Conf. ISAAC 2000. Volume 1969 of LNCS., Springer-Verlag (2000) 432–443
10. Karavelas, M.I., Emiris, I.Z.: Predicates for the planar additively weighted Voronoi diagram. Technical Report ECG-TR-122201-01, INRIA Sophia-Antipolis (2002)
11. Karavelas, M.I., Yvinec, M.: Dynamic additively weighted Voronoi diagrams in 2D. Technical Report No. 4466, INRIA Sophia-Antipolis (2002)
12. Devillers, O.: Improved incremental randomized Delaunay triangulation. In: Proc. 14th Annu. ACM Sympos. Comput. Geom. (1998) 106–115
13. Karavelas, M.: Proximity Structures for Moving Objects in Constrained and Unconstrained Environments. PhD thesis, Stanford University (2001)
14. : The CGAL Reference Manual. (2001) Release 2.3, http://www.cgal.org.
15. Pion, S.: Interval arithmetic: An efficient implementation and an application to computational geometry. In: Workshop on Applications of Interval Analysis to systems and Control. (1999) 99–110