Dynamic Additively Weighted Voronoi Diagrams in 2D

Menelaos I. Karavelas, Mariette Yvinec

INRIA

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Dynamic Additively Weighted Voronoi Diagrams in 2D

Menelaos I. Karavelas — Mariette Yvinec

N° 4466

Mai 2002

THÈME 2

ISSN 0249-6399 ISRN INRIA/RR--4466--FR+ENG

Dynamic Additively Weighted Voronoi Diagrams in 2D

Menelaos I. Karavelas [*] , Mariette Yvinec [†]

Thème 2 — Génie logiciel
et calcul symbolique
Projet PRISME

Rapport de recherche n° 4466 — Mai 2002 — 26 pages

Abstract: In this paper we present a dynamic algorithm for the construction of the additively weighted Voronoi diagram of a set of weighted points on 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 non-empty Voronoi 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 with non-empty Voronoi cell. Deletions can be performed for all sites whether or not their Voronoi 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.

Key-words: additively weighted Voronoi diagram; Delaunay graph; dual graph; dynamic algorithm

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).

[*] INRIA Sophia-Antipolis, Project PRISME, 2004 Route des Lucioles, BP 93, 06902 Sophia-Antipolis Cedex, France, email: Menelaos.Karavelas@sophia.inria.fr

[†] INRIA Sophia-Antipolis, Project PRISME, 2004 Route des Lucioles, BP 93, 06902 Sophia-Antipolis Cedex, France, email: Mariette.Yvinec@sophia.inria.fr

Unité de recherche INRIA Sophia Antipolis
2004, route des Lucioles, BP 93, 06902 Sophia Antipolis Cedex (France)
Téléphone : +33 4 92 38 77 77 — Télécopie : +33 4 92 38 77 65

Diagrammes de Voronoi Additifs Dynamiques en 2D

Résumé : Ce rapport décrit un algorithme dynamique pour construire le diagramme de Voronoï à poids additifs d'un ensemble de points pondérés du plan. L'algorithme proposé représente le diagramme à poids additif à travers son dual. Il est incrémental et pleinement dynamique, c'est à dire permet l'insertion ou la suppression de sites. Une analyse randomisée sur l'ordre d'insertion montre que l'algorithme construit le le diagramme de Voronoï à poids additifs d'un ensemble de n sites parmi lesquelles h ont une cellule de Voronoi non vide, en temps moyen $O(nT(h) + h \log h)$ où $T(k)$ est le temps nécessaire pour répondre à une requête de plus proche voisin (pour une distance à poids additifs) sur un ensemble de k sites à cellules non vides. L'espace mémoire utilisé est $O(n)$. L'algorithme est simple à implémenter et une étude expérimentale laisse présumer un comportement asymptotique en $O(n \log h)$.

Mots-clés : diagramme de Voronoï; triangulation de Delaunay; diagramme de Voronoï à poids additifs; algorithme dynamique

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 with respect to the circle. The 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 :

$$\delta(B_1,B_2) = d(b_1,b_2) - r_1 - r_2. \tag{1}$$

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. This is equivalent to stating that a site is trivial if there exists another site such that the bisector of the two sites does not exist.

The first algorithm for computing the AW-Voronoi diagram appeared in [6]. 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 [14] 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 [17]. 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 [7] 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 $O(n \log n)$. The predicates required in the last two algorithms are rather complicated. Aurenhammer [2] 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. More specifically, a weighted point $P_i = \{(x_i, y_i), w_i\}$ is mapped to a sphere Σ_i with center (x_i, y_i, w_i) and radius $\sqrt{2}w_i$. Then the Voronoi cell of P_i is the projection on the plane of the intersection of the power cell of Σ_i with the upper nappe of the cone of angle $\pi/4$ whose apex is at $(x_i, y_i, -w_i)$. 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* [12], 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 [15, 13]). The algorithm in [13] 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 [10, 11]. 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 [13], 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 [17], 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 [13]. In [13] 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 is to find the nearest neighbor of the new site in the existing AW-Voronoi diagram. Using the nearest neighbor we can then determine easily if the new site is trivial; this is the second stage. If the new site is not trivial we proceed in a way similar to that [13]. Starting from the nearest neighbor of the new site, we find all the Voronoi edges in conflict and then we reconstruct the AW-Voronoi diagram. The search for Voronoi edges in conflict, as well as the reconstruction of the AW-Voronoi diagram are done using the dual graph. This is the third stage of our algorithm. Another novelty of our algorithm is that it permits the deletion of sites, which is not the case in [13].

The remainder of the paper is structured as follows. In Section 2 we give formal definitions for the AW-Voronoi diagram and its dual and review some of the known properties of the AW-Voronoi diagram. We also provide various definitions used in the remainder of the paper. In Section 3 we show how to insert a new site once we know its nearest neighbor in the existing AW-Voronoi. We also discuss how we deal with trivial sites and perform a runtime analysis for our algorithm. In Section 4 we describe how to locate the nearest neighbor of a query site. In Section 5 we describe how to delete sites. In Section 6 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 [9]. Section 7 is devoted to conclusions and directions for further research. In Section A of the Appendix we describe a special embedding of the AW-Delaunay graph.

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

$$V_i = \bigcap_{i \neq j} H_{ij}.$$

The connected set of points that belong to exactly two Voronoi cells are called *Voronoi edges*, whereas points that belong to more than 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 [17, Property 1]. It consists of straight or hyperbolic arcs and each cell is star-shaped with respect to the center of the corresponding site [17, 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, the Voronoi cells of which are empty. 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 (see [17, 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*. Fig. 1(left) shows the AW-Voronoi diagram for a set of 12 sites, among which 2 are trivial.

We call AW-Delaunay graph and note $\mathcal{D}(\mathcal{B})$ 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 [17, 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 [17, 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

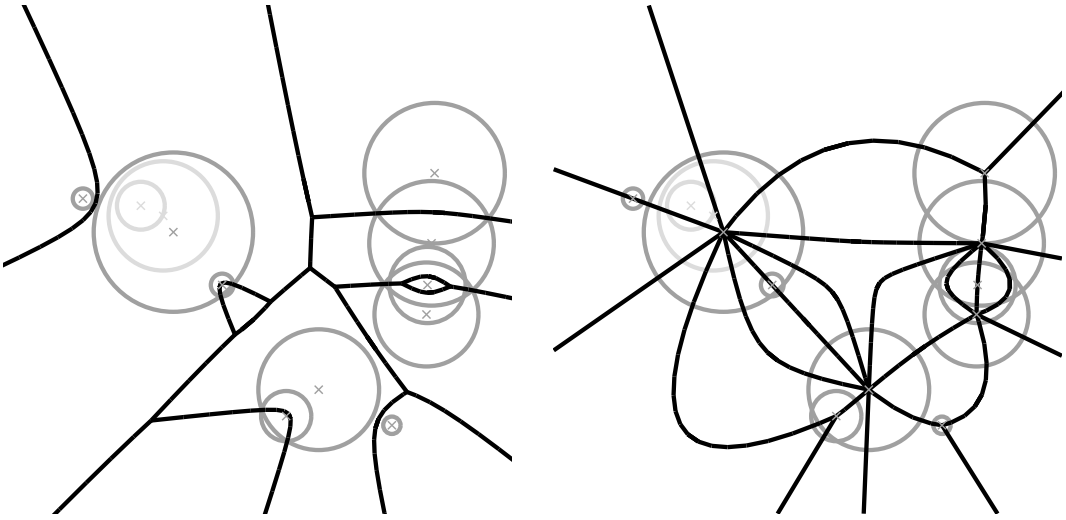

Figure 1: Left: the AW-Voronoi diagram for a set of 12 sites. Non-trivial sites are shown in gray. Trivial sites are shown in light gray. The Voronoi skeleton is shown in black. Right: a planar embedding of the AW-Delaunay graph of the same set of sites. The edges of the AW-Delaunay graph are shown in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

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 $\mathcal{V}_1(\mathcal{B})$ (such an edge occurs for each pair of sites B_i and B_j that appear consecutively on the convex hull of $\mathcal{B}$). 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. Unless otherwise stated, 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. We will show later that our algorithm, using an implicit perturbation scheme, 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*. The following theorem couples the existence of edges in $\mathcal{D}(\mathcal{B})$ with exterior and interior bitangent Voronoi circles of sites in $\mathcal{B}$.

Theorem 1 (Global Property) *There exists an edge connecting B_i and B_j in $\mathcal{D}(\mathcal{B})$ if and only if one of the following holds:*

1. *There exists an exterior bitangent Voronoi circle of B_i and B_j which intersects no site $B_k \in \mathcal{B}$, $k \neq i, j$.*
2. *There exists an interior bitangent Voronoi circle of B_i and B_j, which is contained in no site $B_k \in \mathcal{B}$, $k \neq i, j$.*

Proof. Let e_{ij}^{kl} be an edge of $\mathcal{D}(\mathcal{B})$. Let $\alpha_{ij}^{kl} \in \mathcal{V}(\mathcal{B})$ be the dual edge of e_{ij}^{kl}, which by assumption has non-empty interior. Let y be a point in the interior of α_{ij}^{kl}. Consider the circle C centered at y with radius $|\delta(y, B_i)| = |\delta(y, B_j)|$. Clearly C is a bitangent Voronoi circle of B_i and B_j. Suppose that C is an exterior bitangent Voronoi circle and suppose that C intersects with a third site B_m. If C is not tangent to B_m, then $\delta(y, B_m) < \delta(y, B_i) = \delta(y, B_j)$, which contradicts the assumption that $y \in V_i \cap V_j$. If C is tangent to B_m then y belongs to V_m as well. But this contradicts the fact that y is an interior point of α_{ij}^{kl}. Suppose now that C is an interior bitangent Voronoi circle and suppose it is contained in a third site B_k. If C is tangent to B_m, then y belongs to $V_i \cap V_j \cap V_m$, which contradicts our assumption that y is an interior point of α_{ij}^{kl}. If C is contained in the interior of B_m, then $\delta(y, B_m) < \delta(y, B_i) = \delta(y, B_j)$, which implies that $y \in V_m$ and contradicts the fact that $y \in V_i \cap V_j$. Hence, in both cases, C is the desired bitangent Voronoi circle.

Conversely, assume there exists an exterior bitangent Voronoi circle C of B_i and B_j, that intersects no other site B_k. Let y be the center of C. Then $y \in V_i \cap V_j$, since $\delta(y, B_i) = \delta(y, B_j)$ and $\delta(y, B_i) < \delta(y, B_m)$, for all $m \neq i, j$. Suppose that y is an endpoint of an edge in $\mathcal{V}(\mathcal{B})$. Then there exists a third site B_m that is tangent to C; this contradicts the assumption that C intersects only B_i and B_j. Hence y has to be in the interior of some edge α_{ij}^{kl} of $\mathcal{V}(\mathcal{B})$. But then there exists an edge between B_i and B_j in $\mathcal{D}(\mathcal{B})$. Assume now that C is an interior bitangent Voronoi circle of B_i and B_j, which is contained inside no other site B_m, $m \neq i, j$. Let y be the center of C. Then $\delta(y, B_i) = \delta(y, B_j)$ and since C is contained inside no other site B_m, $m \neq i, j$, we have that $\delta(y, B_m) > \delta(y, B_i)$. Hence $y \in V_i \cap V_j$. If y was an endpoint of $V_i \cap V_j$, then there would be a third site B_m, $m \neq i, j$, with $\delta(y, B_m) = \delta(y, B_i)$. But then C would be contained in B_m, which contradicts our assumption. Hence y is an interior point of $V_i \cap V_j$, which implies that there exists an edge connecting B_i and B_j in $\mathcal{D}(\mathcal{B})$. □

Let B_i, B_j and B_k be three sites, such that no one is contained inside the others. A circle that is tangent to all three of them, that does not contain any of them and is not included in any of them is called an *exterior tritangent Voronoi circle*. A circle that is tangent to all three of them and lies

in $B_i \cap B_j \cap B_k$ is called an *interior tritangent Voronoi circle*. 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 (see [17, Property 5]).

Let P_i, P_j, P_k be the points of tangency of the sites B_i, B_j, B_k with one of their tritangent Voronoi circles. Let also $\texttt{CCW}(\cdot,\cdot,\cdot)$ denote the usual *orientation test* of three points. If $\texttt{CCW}(P_i, P_j, P_k) > 0$ we say that the tritangent Voronoi circle is a CCW-*Voronoi circle* of the triple B_i, B_j, B_k. If $\texttt{CCW}(P_i, P_j, P_k) < 0$, we say that the tritangent Voronoi circle is a CW-*Voronoi circle* of the triple B_i, B_j, B_k. It can be shown that three sites in a given order can have at most one CCW- or CW-Voronoi circle, which can be either exterior or interior (cf. [9]).

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}. Clearly, 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 $b_i b_j$. 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 $\mathbb{R}$. Given a bitangent Voronoi circle C of B_i and B_j, we define $\zeta_{ij}(C)$ to be the parameter value $\zeta_{ij}(c)$, where $c \in \pi_{ij}$ is the center of C. In addition, given a point $c \in \pi_{ij}$, we denote the bitangent Voronoi circle of B_i and B_j centered at c as $W_{ij}(c)$.

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 $\delta(B, W_{ij}(c)) < 0$. 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 e_{ij}^{kl} be an edge of $\mathcal{D}(\mathcal{B})$ that is the dual of an edge α_{ij}^{kl} of $\mathcal{V}(\mathcal{B})$. 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}) and call c_{ijk} (resp. c_{ijl}) the ijk-endpoint or ijk-vertex (resp. ijl-endpoint or ijl-vertex) of α_{ij}^{kl}. 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. $\tilde{R}_{ij}^{kl}(B) = \emptyset$, in which case we say that B *is not in conflict* with α_{ij}^{kl}.
2. $\tilde{R}_{ij}^{kl}(B)$ consists of a single connected interval, in which case we further distinguish between the following cases :
   (a) $\tilde{\alpha}_{ij}^{kl} = \tilde{R}_{ij}^{kl}(B)$, in which case we say that B *is in conflict with the entire edge* α_{ij}^{kl}.
   (b) $\tilde{R}_{ij}^{kl}(B)$ contains ξ_{ijk}, but not ξ_{ijl}, in which case we say that B *is in conflict with the* ijk*-vertex of* α_{ij}^{kl}.
   (c) $\tilde{R}_{ij}^{kl}(B)$ contains ξ_{ijl}, but not ξ_{ijk}, in which case we say that B *is in conflict with the* ijl*-vertex of* α_{ij}^{kl}.
   (d) $\tilde{R}_{ij}^{kl}(B)$ contains neither ξ_{ijk} nor ξ_{ijl}, in which case we say that B *is in conflict with the interior of* α_{ij}^{kl}.

3. $\tilde{R}_{ij}^{kl}(B)$ consists of a two disjoint intervals, including respectively ξ_{ijk} and ξ_{ijl}, in which case we say that B *is in conflict with both vertices 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 and show its correctness. 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 :

1. Locate the nearest neighbor $NN(B)$ of B in $\mathcal{B}_m$, with respect to the distance function (1).
2. Test if B is trivial.
3. Find the conflict region of B and repair the AW-Delaunay graph.

We postpone the discussion on the location of the nearest neighbor until Section 4. The remaining phases of the insertion procedure are discussed in the sequel.

3.1 Triviality test

The first test we have to do is to determine whether B is trivial or not. The following lemma gives an answer to this question.

Lemma 1 *B is trivial if and only if $B \subset NN(B)$.*

Proof. Clearly, if $B \subset NN(B)$, B is trivial. Suppose now that B is trivial but $B \not\subset NN(B)$. Since B is trivial, there exists a non-trivial site $B' \in \mathcal{B}_m$, such that $B \subset B'$. This is equivalent to requiring that

$$\delta(B, B') < -2r,$$

where r is the radius of B. Since $NN(B)$ is the nearest neighbor of B we also have that

$$\delta(B, NN(B)) < \delta(B, B'),$$

which gives

$$\delta(B, NN(B)) < -2r.$$

But the last relation implies that B is contained in $NN(B)$, i.e., we have a contradiction. □

Hence once we have found the nearest neighbor of the new site B we can test in $O(1)$ time whether it is trivial or not.

3.2 Finding the conflict region

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 $\mathcal{V}_1(\mathcal{B})$ and $\partial R_m(B)$ is a set of points on edges of $\mathcal{V}_1(\mathcal{B})$. 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 [13, 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 [13]. The idea is to perform a *depth first search* (DFS) on $\mathcal{V}_1(\mathcal{B})$ 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. The following lemma suggests a way to do this.

Lemma 2 *Let B be a non-trivial site in $\mathcal{B} \setminus \mathcal{B}_m$. Let $NN(B)$ be the nearest neighbor of B in $\mathcal{B}_m$ and let $V_{NN(B)}$ be the Voronoi cell of $NN(B)$ in $\mathcal{V}(\mathcal{B}_m)$. Then B has to be in conflict with at least one of the edges of $V_{NN(B)}$.*

Proof. Since $NN(B)$ is the nearest neighbor of B in $\mathcal{B}_m$, the center b of B must lie in the Voronoi cell $V_{NN(B)}$ of $NN(B)$ in $\mathcal{V}(B_m)$. Suppose that b lies on one of the Voronoi edges α on the boundary of $V_{NN(B)}$. Since, b is contained in the Voronoi cell V_B of B in $\mathcal{V}(\mathcal{B}_{m+1})$, we immediately get that α is in conflict with B. Suppose that b lies in the interior of $V_{NN(B)}$ and assume that B is not in conflict with any of the Voronoi edges on the boundary of $V_{NN(B)}$. Then V_B must lie in the interior of $V_{NN(B)}$. This however contradicts the fact that $V_{NN(B)}$ is simply connected. Thus, the assumption that B is not in conflict with any of the edges on the boundary of $V_{NN(B)}$ is false. □

Since B has to be in conflict with at least one of the edges of the Voronoi cell $V_{NN(B)}$ of $NN(B)$, we 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 cases :

- v' has not been marked. We distinguish between the following two cases.

  - B is in conflict with the entire edge α :
    We replace α in L by the remaining Voronoi edges adjacent to v', in counter-clockwise order. We then continue recursively on v'.
  - B is not in conflict with the entire edge α :
    We have reached a Voronoi edge adjacent to $\partial R_m(B)$. The list L remains unchanged and the DFS backtracks.

- v' has already been marked. We distinguish between the following two cases.

  - B is in conflict with the entire edge α :
    This means that have found a cycle in $R_m(B)$, or equivalently, B contains a site in $\mathcal{B}_m$, which will become trivial. Since α belongs to L, but it does not contain any points of $\partial R_m(B)$, we remove it from L. The DFS then backtracks.
  - B is not in conflict with the entire edge α :
    We have that 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.

Lemma 3 *The search algorithm described above finds the boundary $\partial R_m(B)$ of the conflict region $R_m(B)$, i.e., at the end of the algorithm $L = \partial R_m(B)$.*

Proof. If $R_m(B)$ is the interior of an edge of $\mathcal{V}(\mathcal{B}_m)$, then we discover $\partial R_m(B)$ once we find the first conflict of B with the Voronoi edges of the boundary of $V_{NN(B)}$. If $R_m(B)$ is not the interior of some edge of $\mathcal{V}(\mathcal{B}_m)$, every point w on $\partial R_m(B)$ has to be contained in a Voronoi edge adjacent to a Voronoi vertex v that belongs to $R_m(B)$. Suppose that our algorithm did not find a vertex w in $\partial R_m(B)$. This implies that it did not reach the vertex v associated with w. Since $R_m(B)$ is connected (cf. [13, Lemma 1]), our DFS search algorithm will discover all vertices of $R_m(B)$. This contradicts the fact that v was not reached. □

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 can be done by using the duality between the AW-Voronoi diagram and the AW-Delaunay graph. This duality maps vertices to faces and edges to edges. A Voronoi vertex is mapped to a triangle in the AW-Delaunay graph and a Voronoi edge to a Delaunay edge in the dual graph. In this context, when we say that B is in conflict with a Delaunay edge e we mean that B is in conflict with its dual Voronoi edge α. The type of conflict of B with e is the type of conflict of B with α. Similarly, when we say that B is in conflict with a triangle t in $\mathcal{D}(\mathcal{B}_m)$, we mean that B is in conflict with its dual Voronoi vertex v in $\mathcal{V}(\mathcal{B}_m)$. Under the duality mapping a point on $\partial R_m(B)$ can be represented by the dual edge e of the Voronoi edge α that contains it. We call $\partial^* R_m(B)$ the set of all Delaunay edges whose dual edge contains a point in $\partial R_m(B)$. The list L is now a list of Delaunay edges. At the end of the search algorithm L will

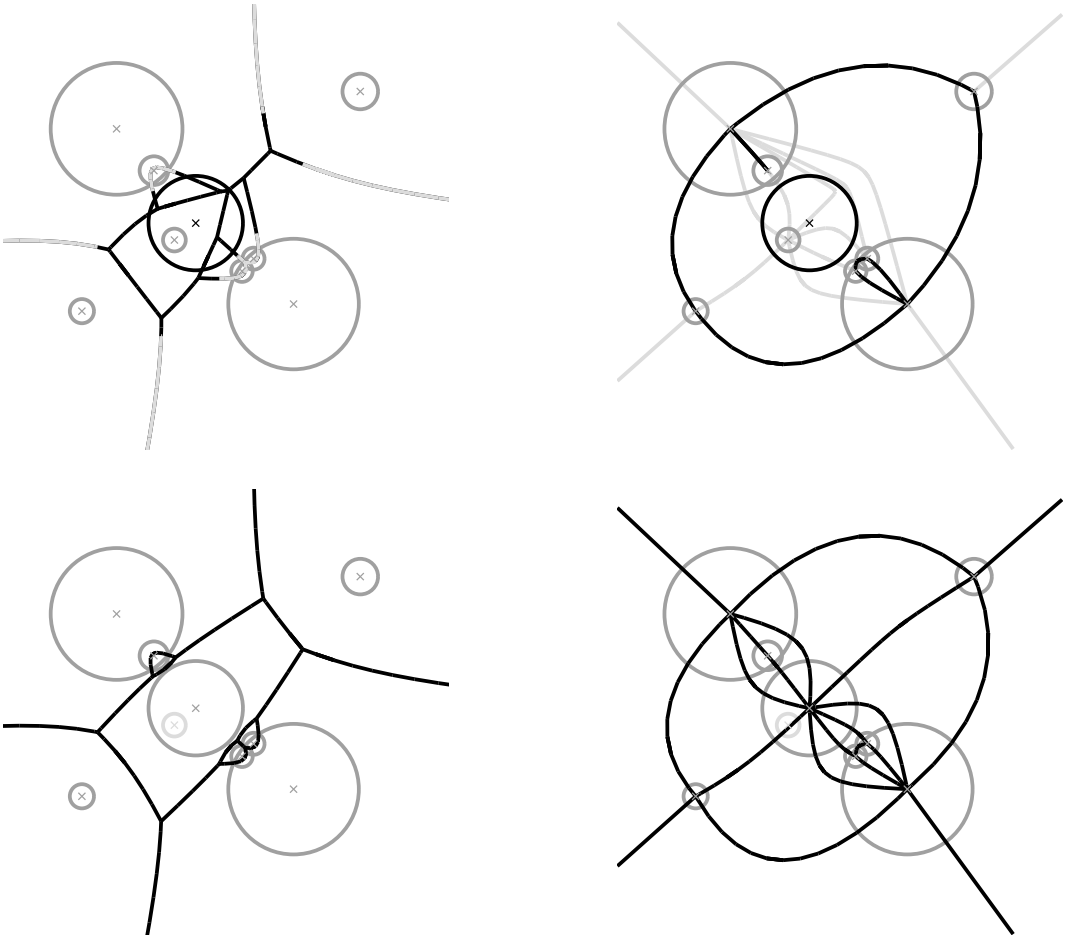

Figure 2: Top left: The AW-Voronoi diagram for a set of sites (gray) and the conflict region (black) of a new site (also black). The portion of the Voronoi skeleton that does not belong to the conflict region of the new site is shown in light gray. Top right: The AW-Delaunay graph for the same set of sites and the set $\partial^* R$ of the new site (black). Although $\partial^* R$ is a simple polygon, it contains an edge twice and two vertices twice. Bottom left: 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. Bottom right: The AW-Delaunay graph after the insertion of the new site.

be equal to $\partial^* R_m(B)$, which is in fact the boundary of the star of B in $\mathcal{D}(\mathcal{B}_{m+1})$. Inserting B in $\mathcal{D}(\mathcal{B}_m)$ then reduces to computing L and staring the hole represented by L using B. The search

algorithm in terms of the dual graph can be stated as follows. To find the first conflict of B with the Voronoi edges of $V_{NN(B)}$, we look at the Delaunay edges adjacent to $NN(B)$ in $\mathcal{D}(\mathcal{B}_m)$. Let e be the first Delaunay edge found to be in conflict with B. If B is in conflict with the interior of e we have found $\partial^* R_m(B)$ and L contains two copies of e with different orientations. If B is not in conflict with the interior of e, it has to be in conflict with at least one of the two triangles in $\mathcal{D}(\mathcal{B}_m)$ adjacent to e. Let t be this triangle. We put all three Delaunay edges of t in L in counter-clockwise order. The DFS will then visit recursively all triangles in $\mathcal{D}(\mathcal{B}_m)$ in conflict with B. Suppose that we have arrived at a triangle t. At first, we mark t. Then we look at all the triangles t' adjacent to t. Let e be the common edge of t and t'. We distinguish between the following cases :

- t' has not been marked. Then consider the following two cases.
  - B is in conflict with the entire edge e :
    We replace e in L by the remaining two edges of t' in counter-clockwise order. We then continue recursively on t'.
  - B is not in conflict with the entire edge e :
    We have reached one of the edges of $\partial^* R_m(B)$. The list L remains unchanged and the DFS backtracks.
- t' has been marked. Then consider the following two cases.
  - B is in conflict with the entire edge e :
    We remove e from L and the DFS backtracks.
  - B is not in conflict with the entire edge e :
    Then $e \in \partial^* R_m(B)$ and the DFS backtracks. In this case, B is conflict with both vertices of e, i.e., e appears in $\partial^* R_m(B)$ twice (with different orientations).

The set $\partial^* R_m(B)$ of a site B is shown in Fig. 2(top right). Note, that $\partial^* R_m(B)$ is a simple polygon that can contain both a vertex or an edge multiple times. Multiple vertices appear in $\partial^* R_m(B)$ when there exist multiple Voronoi edges between the new site and an old one. Multiple edges appear in $\partial^* R_m(B)$, when the new site is adjacent to an old site that has degree 2 in $\mathcal{D}(\mathcal{B}_{m+1})$.

Remark: 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.

The pseudo-code for the recursive procedure that constructs $\partial^* R_m(B)$ by performing a DFS on the dual AW-Delaunay graph is given below.

EXPANDCONFLICTREGION(t, L, B)
1: Mark t
2: **for all** edges e^α of t **do**
3: $t' \leftarrow$ triangle adjacent to t through e^α

```
    if t' has not been marked then
      if B is in conflict with the entire Voronoi edge α then
        replace e^α in L by the other two edges of t'
        EXPANDCONFLICTREGION(t', L, B)
      end if
    else {t' has been marked}
      if B is in conflict with the entire edge α then
        remove e^α from L
      end if
    end if
  end for
```

3.3 Storing the trivial 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. One approach to treat trivial sites is to totally discard of them. However, 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 a list for every non-trivial site, which contains all trivial sites that have the non-trivial site as their parent. In the sequel of this subsection, we describe how to maintain these lists as we insert new sites.

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 $L_{tr}(NN(B))$. In this case $\mathcal{L}_{m+1}$ is constructed in constant time from $\mathcal{L}_m$. 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 $L_{tr}(B)$.

3.4 Runtime analysis

If we implement the lists $L_{tr}(\cdot)$ as doubly-linked lists, moving all sites in $L_{tr}(B_1)$ to $L_{tr}(B_2)$ for some non-trivial sites B_1, B_2 can be done by simply appending $L_{tr}(B_1)$ to $L_{tr}(B_2)$; this in turn can be done in constant time. Therefore, the time to construct $\mathcal{L}_{m+1}$ from $\mathcal{L}_m$ is $O(|\mathcal{B}_m^-(B)|)$. Since $|\mathcal{B}_m^-(B)| = O(|R_m(B)|)$ we conclude that the time to construct $\mathcal{L}_{m+1}$ is $O(|R_m(B)|)$.

Therefore, the running time for constructing the AW-Voronoi diagram $\mathcal{V}(\mathcal{B}_{m+1})$ and the set $\mathcal{L}_{m+1}$, once we have found the first conflict, is proportional to the complexity $|R_m(B)|$ of $R_m(B)$.

Finding the first conflict can be done in time $O(\log d_{NN(B)})$, where $d_{NN(B)}$ is the degree of $NN(B)$ in $\mathcal{B}_m$. Consider the set of directions with respect to the center of $NN(B)$ defined by Voronoi vertices of the cell of $NN(B)$. These directions subdivide the interval $[0, 2\pi)$ in $O(d_{NN(B)})$ angular sections. We can perform a binary search to determine which sector contains b. If B is non-trivial, is has to be in conflict with the edge corresponding to the angular section in which b resides. Therefore, the time to locate the first conflict is $O(\log d_{NN(B)})$. Hence,

Lemma 4 *Let $\mathcal{B}_m$ be a subset of $\mathcal{B}$ for which we have constructed the AW-Voronoi diagram. Let B a site in $\mathcal{B} \setminus \mathcal{B}_m$ and let $\mathcal{B}_{m+1} = \mathcal{B}_m \cup \{B\}$. Given the nearest neighbor $NN(B)$ of B in $\mathcal{B}_m$:*

1. *We can determine if B is trivial in time $O(1)$.*
2. *We can find the first conflict of B with $\mathcal{V}_1(\mathcal{B}_m)$ in time $O(\log d_{NN(B)})$.*
3. *We can construct $\mathcal{V}(\mathcal{B}_{m+1})$ and $\mathcal{L}_{m+1}$ in time $O(|R_m(B)|)$.*

Suppose that $\mathcal{B}$ consists of n sites among which h are non-trivial. Let $T(k)$ denote the time to find the nearest neighbor of a query site within a set of non-trivial sites of size k. The total time for the construction of the AW-Voronoi diagram is the time $T_1(n, h)$ to find the nearest neighbors and detect the trivial sites plus the time $T_2(h)$ to find the first conflicts plus the time $T_3(h)$ to update the AW-Voronoi diagram. Clearly, $T_1(n, h) = O(nT(h))$ worst case and, by Lemma 4, $T_2(h) = O(h \log h)$ worst case. By Lemma 4 again, the cost of adding a new site is proportional to the number of Voronoi edges destroyed by the new site. Hence, $T_3(h)$ is proportional to the total number of Voronoi edges destroyed during the course of our algorithm. Applying a randomized analysis similar to that in [3, Chapter 5], we can easily deduce that $T_3(h) = O(h)$ in the expected sense, where the expectation is on the insertion order of sites. Hence the total expected running time of the algorithm presented is $O(nT(h) + h \log h)$. Thus,

Theorem 2 *Let $\mathcal{B}$ be a set of n sites among which h are non-trivial. The total expected running time for constructing the AW-Voronoi diagram with the incremental algorithm presented is $O(nT(h) + h \log h)$.*

4 Nearest neighbor location

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 pseudo-code for the nearest neighbor procedure just described is presented below.

NEARESTNEIGHBOR($B, \mathcal{B}_m$)
1: Select a site $B' \in \mathcal{B}_m$ arbitrarily
2: **repeat**

```
    f_2 ← false
    for all neighbors B'' of B' in D(B_m) do
      if δ(B, B'') < δ(B, B') then
        B' ← B''
        f_2 ← true
        break
      end if
    end for
  until f_2 ≠ true
  return B'
```

Lemma 5 *The procedure* NEARESTNEIGHBOR$(B, \mathcal{B}_m)$ *finds the nearest neighbor of* B *in* $\mathcal{B}_m$.

Proof. Let B' be the site returned as the nearest neighbor of B by the procedure described above. Let $\mathcal{B}'_m$ be the set of all neighbors of B' in $\mathcal{D}(\mathcal{B}_m)$. By construction we have that

$$\delta(b, B_i) \geq \delta(b, B'), \quad \forall B_i \in \mathcal{B}'_m. \tag{2}$$

Consider the AW-Voronoi diagram of the set $\mathcal{B}''_m = \mathcal{B}'_m \cup \{B'\}$. Clearly, the Voronoi cells of B' in $\mathcal{V}(\mathcal{B}_m)$ and $\mathcal{V}(\mathcal{B}''_m)$ coincide.

Suppose that $B' \neq NN(B)$. This implies that b is not inside the Voronoi cell of B' in $\mathcal{V}(\mathcal{B}_m)$, and thus b is not inside the Voronoi cell of B' in $\mathcal{V}(\mathcal{B}''_m)$. In turn, this implies that b belongs to the Voronoi cell in $\mathcal{V}(\mathcal{B}''_m)$ of some site $B'' \in \mathcal{B}'_m$, i.e., $\delta(b, B'') < \delta(b, B')$. This, however, contradicts relation (2). Hence, $B' = NN(B)$. □

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 [5] for the Delaunay triangulation for points. The method consists of building a hierarchical set of AW-Delaunay graphs $\{\mathcal{D}_i\}_{i=0}^{K}$. The graph $\mathcal{D}_0$ is the AW-Delaunay graph of the whole set of sites. Then, each site inserted in the graph $\mathcal{D}_i$ is inserted in the graph $\mathcal{D}_{i+1}$ with probability $1/\beta$, where $\beta > 0$ is a parameter. Thus, each site $B \in \mathcal{B}$ belongs to all $\mathcal{D}_j$, for $i \leq j$ with probability $1/\beta^i$, which implies that the expected height of the AW-Delaunay hierarchy is $O(\log_\beta h)$. The location of the nearest neighbor of a query site B_q is done successively at each level using the procedure NEARESTNEIGHBOR. However, at each level (except the highest one) instead of using an arbitrary site as a starting point for the nearest neighbor search, we use the nearest neighbor found at level $i + 1$. Once we have located $NN(B_q)$, we can check if B_q is trivial and perform the insertion of B_q as usual in the AW-Delaunay graph $\mathcal{D}_0$. The level i of B_q is then randomly chosen and B_q is inserted in all $\mathcal{D}_j$ for $0 < j \leq i$. There is one final detail we need to take care of. It is possible that B_q includes sites that appear in levels larger than i. In this case we must remove all such sites from all levels they appear in.

The randomized time analysis for the location and insertion of a point in the Delaunay hierarchy has been given in [5]. 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 6).

5 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. We distinguish between the cases where B is non-trivial or trivial.

5.1 Deleting a non-trivial site

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 $L_{tr}^-(B) = L_{tr} \setminus L_{tr}^+(B)$, $\mathcal{B}_s = \mathcal{B}_\gamma \cup L_{tr}(B)$ and $\mathcal{B}_s^+ = \mathcal{B}_\gamma \cup L_{tr}^+(B)$. The main idea is given by the following lemma.

Lemma 6 *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 $\mathcal{B}_s^+$.*

Proof. Consider a point $p \in V_B$. If $L_{tr}(B) = \emptyset$, then the second nearest neighbor of p is one of the sites in $\mathcal{B}_\gamma$ (p is contained in the Voronoi cell of one of the sites in $\mathcal{B}_\gamma$). If $L_{tr}(B) \neq \emptyset$, then the second nearest neighbor of B can only be influenced by the sites in $L_{tr}^+(B)$. Hence, the second nearest neighbor of p has to be one of the sites in $\mathcal{B}_s^+$.

Let $B_t \in L_{tr}^-(B)$. If B_t is inside one of the sites in $L_{tr}^+(B)$ we are done. Otherwise, let $\mathcal{B}'$ be the subset of non-trivial sites of $\mathcal{B} \setminus (\{B\} \cup L_{tr}^+(B))$ that contain B_t. For each site $B' \in \mathcal{B}'$, consider the interior bitangent Voronoi circle $C_{B,B'}$ that contains B_t, whose radius $r_{B,B'}$ is maximal. Among all sites in $\mathcal{B}'$, let B'_{max} be the one for which $r_{B,B'}$ is maximized. If B'_{max} is a neighbor of B in $\mathcal{V}(\mathcal{B})$ we are done. Otherwise, there must exist a non-trivial site $B'' \in \mathcal{B} \setminus (\{B\} \cup L_{tr}^+(B))$ such that $C_{B,B'} \subset B''$. Clearly, $B_t \subset B''$, i.e., $B'' \in \mathcal{B}'$. It is also easy to verify that $r_{B,B''} > r_{B,B'_{max}}$, which contradicts the fact that $r_{B,B'_{max}}$ is maximal. □

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 Subsection 3.2. Let $\partial^* R_s$ be the representation, by means of the dual edges, of the conflict region of B in $\mathcal{B}_s$. The triangles inside $\partial^* R_s$ 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 $\mathcal{L}(\mathcal{B} \setminus \{B\})$.

5.2 Deleting a trivial site

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 $L_{tr}(B')$. By Lemma 1, $B \subset NN(B)$. Hence B must be in the list $L_{tr}(B')$ 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 [8, Chapter 5]. Hence the deletion of a trivial site can be done as follows :

1. Find the nearest neighbor $NN(B)$ of B;
2. 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 $B \in L_{tr}(B')$;
3. Once the site B', such that $B \in L_{tr}(B')$, is found, delete B from $L_{tr}(B')$.

5.3 Runtime analysis

In the case where B is non-trivial, the cost of the deletion procedure is the cost to construct $\mathcal{V}(\mathcal{B}_s)$ plus the cost to *retriangulate* the star of B in $\mathcal{D}(\mathcal{B})$ plus the cost to create $\mathcal{L}(\mathcal{B} \setminus \{B\})$ from $\mathcal{L}(\mathcal{B})$. By Theorem 2, it takes $O(|\mathcal{B}_s| T(|\mathcal{B}_s^+|) + |\mathcal{B}_s^+| \log |\mathcal{B}_s^+|)$ expected time to construct $\mathcal{V}(\mathcal{B}_s)$. We can also retriangulate the star of B and construct $\mathcal{L}(\mathcal{B} \setminus \{B\})$ from $\mathcal{L}(\mathcal{B})$ in time linear to the complexity of $\mathcal{V}(\mathcal{B}_s)$, i.e., in time $O(|\mathcal{B}_s^+|)$.

If B is trivial, then we need to find $NN(B)$, which takes $T(h)$ time. Then we need to search the connected component $\mathcal{C}$ in which $NN(B)$ belongs and search all the lists $L_{tr}(B')$ for all $B' \in \mathcal{C}$ that contain B. This clearly takes $O(n)$ time, in the worst case. The deletion of B from the correct list takes $O(1)$ time. Hence the total time to delete a trivial site B is $O(n)$. Summarizing :

Theorem 3 *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. Then :*

1. *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.*
2. *If B is trivial and $B \in L_{tr}(B')$, for some $B' \in \mathcal{B}$, B can be deleted from $\mathcal{L}(\mathcal{B})$ in worst case time $O(n)$.*

Remark: Although the bound $O(n)$ for the time to delete a trivial site is tight in the worst case, it is a very pessimistic one and it is attained for very specific input sets and under very specific insertion orders.

6 Predicates and implementation

6.1 Predicates

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 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 [9], where is it also shown that its algebraic degree is 12.

4. 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 [9]. The corresponding algebraic degree is shown to be 16 in the input quantities, using techniques from Sturm sequences theory.

6.2 Implementation

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 procedure NEAREST-NEIGHBOR. (see beginning of Section 4). The second implementation maintains a hierarchy of AW-Delaunay graphs. The nearest neighbor location is done successively at each level, by means of the procedure NEARESTNEIGHBOR, using as starting point the nearest neighbor found at the previous level. The predicates are evaluated exactly and they have been implemented using two scenarios. The first one is adapted to number types that support the operations $+$, $-$, $\times$, $/$ and $\sqrt{\ }$ exactly. The second one requires that only the operations $+$, $-$ and $\times$ are done exactly. Both algorithms were implemented in C++, following the design of the library CGAL [4]. The implementations of both algorithms are expected to become part of the CGAL distribution in the near future.

6.3 Benchmarks

Computations with inexact number types such as `double` of C++ can often produce undesired behavior in geometric algorithms due to wrong results in the computation of the geometric predicates

of the algorithm. These results are caused by the accumulation of rounding errors during the evaluation of the predicates. Exact numbers types on the other hand provide the necessary robustness but are very costly. A trade-off between the two solutions is to use filtering techniques, the discussion of which is beyond the scope of this paper. In this context, we chose to perform the benchmarks using the interval arithmetic package of CGAL [16], which performs dynamic filtering. The interval arithmetic package is parameterized by an inexact and an exact number type. If the inexact number type is not sufficient in order to compute the predicate correctly, computations are done using the exact number type. In our experiments we use as inexact number type the `double` of C++ and as exact number type `MP_Float` of CGAL [4]. We also chose to use the implementations of our algorithms that do not use square roots for the computations of the predicates.

The two algorithms were tested on random circle sets of size $n \in \{10^3, 10^4, 10^5, 10^6\}$. 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 one of the values in the set $\{1.00, 0.95, 0.80, 0.50\}$. T_1 denotes the running time of the algorithm that uses only one level of the AW-Delaunay graph, and T_2 denotes the running time of the algorithm that uses the AW-Delaunay hierarchy. The last two columns of Table 1 have been added for convenience. They show the ratios of the running times of the two algorithms over the quantity $n \log h$. For our experiments we used a PC with Pentium-III 1GHz running Linux.

It is clear from the last two columns 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 definitely not a good choice for data sets where n is large and $h = \Theta(n)$.

7 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 locate 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 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

n	h	h/n	T_1	T_2	$T_1/(n \log h)$	$T_2/(n \log h)$
1 000	1 000	1.00	0.33	0.33	1.10×10^{-4}	1.10×10^{-4}
1 000	949	0.95	0.35	0.35	1.18×10^{-4}	1.24×10^{-4}
1 000	797	0.80	0.33	0.37	1.14×10^{-4}	1.27×10^{-4}
1 000	504	0.50	0.26	0.29	0.96×10^{-4}	1.07×10^{-4}
10 000	10 000	1.00	4.75	3.59	1.18×10^{-4}	0.90×10^{-4}
10 000	9 454	0.95	4.63	3.77	1.16×10^{-4}	0.95×10^{-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	94 570	0.95	87.29	40.11	1.75×10^{-4}	0.81×10^{-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	950 008	0.95	> 36 min	446.28	–	0.75×10^{-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.

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.

Acknowledgments

The authors would like to thank Jean-Daniel Boissonat for fruitful discussions.

References

[1] N. Amenta, S. Choi, and R. K. Kolluri. The power crust, unions of balls, and the medial axis transform. *Comput. Geom. Theory Appl.*, 19:127–153, 2001.

[2] F. Aurenhammer. Power diagrams: properties, algorithms and applications. *SIAM J. Comput.*, 16:78–96, 1987.

[3] Jean-Daniel Boissonnat and Mariette Yvinec. *Algorithmic Geometry*. Cambridge University Press, UK, 1998. Translated by Hervé Brönnimann.

[4] *The CGAL Reference Manual*, 2.3 edition, 2001. http://www.cgal.org.

[5] Olivier Devillers. Improved incremental randomized Delaunay triangulation. In *Proc. 14th Annu. ACM Sympos. Comput. Geom.*, pages 106–115, 1998.

[6] R. L. Drysdale, III and D. T. Lee. Generalized Voronoi diagrams in the plane. In *Proc. 16th Allerton Conf. Commun. Control Comput.*, pages 833–842, 1978.

[7] S. Fortune. A sweepline algorithm for Voronoi diagrams. In *Proc. 2nd Annu. ACM Sympos. Comput. Geom.*, pages 313–322, 1986.

[8] Menelaos Karavelas. *Proximity Structures for Moving Objects in Constrained and Unconstrained Environments.* PhD thesis, Stanford University, 2001.

[9] Menelaos I. Karavelas and Ioannis Z. Emiris. Predicates for the planar additively weighted Voronoi diagram. Technical Report ECG-TR-122201-01, INRIA Sophia-Antipolis, 2002.

[10] D.-S. Kim, D. Kim, and K. Sugihara. Voronoi diagram of a circle set constructed from Voronoi diagram of a point set. In D. T. Lee and S.-H. Teng, editors, *Proc. 11th Inter. Conf. ISAAC 2000*, volume 1969 of *LNCS*, pages 432–443. Springer-Verlag, 2000.

[11] D.-S. Kim, D. Kim, K. Sugihara, and J. Ryu. Robust and fast algorithm for a circle set Voronoi diagram in a plane. In V. N. Alexandrov et al., editor, *Proceedings of the 2001 International Conference on Computational Science*, volume 2073 of *LNCS*, pages 718–727. Springer-Verlag, 2001.

[12] Rolf Klein. *Concrete and Abstract Voronoi Diagrams*, volume 400 of *Lecture Notes Comput. Sci.* Springer-Verlag, 1989.

[13] Rolf Klein, Kurt Mehlhorn, and Stefan Meiser. Randomized incremental construction of abstract Voronoi diagrams. *Comput. Geom. Theory Appl.*, 3(3):157–184, 1993.

[14] D. T. Lee and R. L. Drysdale, III. Generalization of Voronoi diagrams in the plane. *SIAM J. Comput.*, 10:73–87, 1981.

[15] K. Mehlhorn, S. Meiser, and C. Ó'Dúnlaing. On the construction of abstract Voronoi diagrams. *Discrete Comput. Geom.*, 6:211–224, 1991.

[16] Sylvain Pion. Interval arithmetic: An efficient implementation and an application to computational geometry. In *Workshop on Applications of Interval Analysis to systems and Control*, pages 99–110, 1999.

[17] Micha Sharir. Intersection and closest-pair problems for a set of planar discs. *SIAM J. Comput.*, 14:448–468, 1985.

A The dual graph

In this section we present a planar embedding for the AW-Delaunay graph, which is an alternative to the embedding presented in [17]. In particular, what we show in this section is that the AW-Voronoi diagram of the tritangent Voronoi circles, augmented with some additional vertices is an embedding of the AW-Delaunay graph of the given set of site. Our result is similar in nature to that in [1] connecting the Euclidean Voronoi diagram and the Delaunay triangulation of a point set through the power diagram of the Voronoi circles of the point set. Our embedding is canonical in contrast to the embedding in [17]. Since trivial sites do not contribute to the AW-Voronoi diagram we assume in this section that we only have non-trivial sites.

As previously, we note $\mathcal{B}$ the set of sites, $\mathcal{V}(\mathcal{B})$ the AW-Voronoi diagram of $\mathcal{B}$ and $\mathcal{D}(\mathcal{B})$ the dual AW-Delaunay graph. As usual we consider here the compatifeid version of the AW-Voronoi diagram and AW-Delaunay graph. Let $\mathcal{C}$ be the set of tritangent Voronoi circles associated with the vertices of $\mathcal{V}(\mathcal{B})$. We consider exterior tritangent circles in $\mathcal{C}$ as weighted points with a positive weight, equal to their radius, whereas interior tritangent Voronoi circles are considered as weigthed points with a negative weight, with absolute value equal to their radius. Consider the AW-Voronoi diagram $\mathcal{V}(\mathcal{C})$ of $\mathcal{C}$. The following lemma relates the centers of sites in $\mathcal{B}$ with the Voronoi diagram of $\mathcal{C}$.

Lemma 7 *Let B be a non-trivial site in $\mathcal{B}$, and let $\mathcal{C}_\gamma(B)$ be the set of tritangent Voronoi circles of the faces of $\mathcal{D}(\mathcal{B})$ adjacent to B. Then, the center b of B lies on the the boundary of the Voronoi cells in $\mathcal{V}(\mathcal{C})$ of all the C_i's in $\mathcal{C}_\gamma(B)$. Moreover the cardinality of $\mathcal{C}_\gamma(B)$ is at least 2.*

1. *If $|\mathcal{C}_\gamma(B)| = 2$, then b lies in the interior of a Voronoi edge in $\mathcal{V}(\mathcal{C})$.*
2. *If $|\mathcal{C}_\gamma(B)| > 2$, then b is a Voronoi vertex in $\mathcal{V}(\mathcal{C})$.*

Proof. The cardinality of $\mathcal{C}_\gamma(B)$ is at least 2 because, as explained in section sec:prelim, each vertex in the compatified version $\mathcal{D}(\mathcal{B})$ of the AW-Delaunay graph has degree at least two. Consider the Voronoi circles $C_i \in \mathcal{C}_\gamma(B)$. Clearly :

$$\delta(B, C_i) = 0, \qquad \forall C_i \in \mathcal{C}_\gamma(B),$$

and

$$\delta(B, C_i) > 0, \qquad \forall C_i \in \mathcal{C} \setminus \mathcal{C}_\gamma(B).$$

We deduce that b of B lies on the boundary of the Voronoi cells in $\mathcal{V}(\mathcal{C})$ of all the C_i's in $\mathcal{C}_\gamma(B)$. If the cardinality of $\mathcal{C}_\gamma(B)$ is 2 then b is contained in the interior of an edge of $\mathcal{V}(\mathcal{C})$. If $|\mathcal{C}_\gamma(B)| > 2$, then b lies on the boundary of at least three Voronoi cells in $\mathcal{V}(\mathcal{C})$, thus it is a Voronoi vertex in $\mathcal{V}(\mathcal{C})$. □

Let $S_2(\mathcal{B})$ be the set of centers of all the vertices of degree 2 in $\mathcal{D}(\mathcal{B})$. By Lemma 7, the points in $S_2(\mathcal{B})$ lie in the interior of Voronoi edges of $\mathcal{V}(\mathcal{C})$. We note $\mathcal{V}^*(\mathcal{C})$ the diagram obtained from $\mathcal{V}(\mathcal{C})$ when edges of $\mathcal{V}(\mathcal{C})$ are splitted with additionnal vertices located at points of $S_2(\mathcal{B})$.

Theorem 4 *Let $\mathcal{B}$ be a set of sites and $\mathcal{C}$ the set of tritangent Voronoi circles of $\mathcal{V}(\mathcal{B})$. Let $S_2(\mathcal{B})$ be the set of vertices of $\mathcal{D}(\mathcal{B})$ of degree 2 and let $\mathcal{V}^*(\mathcal{C})$ be the AW-Voronoi diagram of $\mathcal{C}$ augmented by the vertices in $S_2(\mathcal{B})$. Then $\mathcal{V}^*(\mathcal{C})$ is a valid embedding of the AW-Delaunay graph $\mathcal{D}(\mathcal{B})$ of $\mathcal{B}$.*

Proof. Let B be a site in $\mathcal{B}$ and consider the set $\mathcal{C}_\gamma(B)$ of Voronoi circles corresponding to the faces of $\mathcal{D}(\mathcal{B})$ adjacent to B. By Lemma 7, if $|\mathcal{C}_\gamma| = 2$, then b is contained in the interior of an edge of $\mathcal{V}(\mathcal{C})$. However, by means of the construction of $\mathcal{V}^*(\mathcal{C})$, b is a vertex in $\mathcal{V}^*(\mathcal{C})$. If $|\mathcal{C}_\gamma| > 2$ is greater than 2, then b is a Voronoi vertex in $\mathcal{V}(\mathcal{C})$ and thus b is also a Voronoi vertex in $\mathcal{V}^*(\mathcal{C})$. Hence, all the centers of sites in $\mathcal{B}$ are vertices in $\mathcal{V}^*(\mathcal{C})$.

Consider an edge e_{ij}^{kl} in $\mathcal{D}(\mathcal{B})$ that is the dual of a Voronoi edge α_{ij}^{kl} in $\mathcal{V}(\mathcal{B})$. Let π_{ij} denote the bisector of the sites B_i and B_j. Let C_{ijk} and C_{ijl} be the tritangent Voronoi circles associated with the endpoints of e_{ij}^{kl}. Consider the quad Q_{ijkl} defined by the centers b_i, b_j, c_{ijk}, c_{ijl} of B_i, B_j, C_{ijk}, C_{ijl}. Q_{ijkl} is contained in the union $V_i \cup V_j$ of the Voronoi cells of B_i and B_j in $\mathcal{V}(\mathcal{B})$, but also in the union $V_{ijk}^* \cup V_{ijl}^*$ of the Voronoi cells of C_{ijk} and C_{ijl} in $\mathcal{V}^*(\mathcal{C})$. Since Voronoi cells are star-shaped, Q_{ijkl} does not contain any centers of sites in $\mathcal{B}$ or centers of circles in $\mathcal{C}$, except b_i, b_j, c_{ijk} and c_{ijl}. In particular, the set $\mathcal{Q}$ of all quads defined this way forms a partition of the entire plane. Let σ_{ijkl} be the bisector of C_{ijk} and C_{ijl}. Clearly, σ_{ijkl} passes through the centers b_i and b_j of B_i and B_j. Let β be the arc of σ_{ijkl} delimited by b_i and b_j. Since Voronoi cells are star-shaped, β is contained in Q_{ijkl}. Suppose that β is not an edge of $\mathcal{V}^*(\mathcal{C})$. This implies that β contains a vertex v of $\mathcal{V}^*(\mathcal{C})$. Let V^* be the Voronoi cell in $\mathcal{V}^*(\mathcal{C})$, distinct from V_{ijk}^* and V_{ijl}^*, to which v belongs. Since the Voronoi cells are connected, the Voronoi cell V^* must be entirely contained in Q_{ijkl}. This implies that the center c of the tritangent Voronoi circle C to which V^* corresponds has to be inside Q_{ijkl} as well. This is, however, impossible since Q_{ijkl} is empty of centers of both the original sites as well as the tritangent Voronoi circles. Hence β corresponds to an edge of $\mathcal{V}^*(\mathcal{C})$.

Suppose now that there exists a vertex v of $\mathcal{V}^*(\mathcal{C})$ that does not belong in $\mathcal{B}$. Since the set $\mathcal{Q}$ of quads forms a partition of the plane, v has to be inside a quad Q_{ijkl}. But we have just proved that Q_{ijkl} corresponds to an edge of $\mathcal{V}^*(\mathcal{C})$ and thus is it empty of vertices of $\mathcal{V}^*(\mathcal{C})$. Similarly, there cannot exist an edge e of $\mathcal{V}^*(\mathcal{C})$ other than the ones we have already discovered, since such an edge would have to cut at least one of the quads Q_{ijkl} in $\mathcal{Q}$, which are also empty of edges of $\mathcal{V}^*(\mathcal{C})$.

Hence there exists an 1–1 correspondence between the vertices and edges in $\mathcal{D}(\mathcal{B})$ and $\mathcal{V}^*(\mathcal{C})$. Therefore, $\mathcal{V}^*(\mathcal{C})$ is isomorphic to $\mathcal{D}(\mathcal{B})$, and in particular, it is a planar embedding of $\mathcal{D}(\mathcal{B})$. □

A special case of the above theorem is when the set $\mathcal{B}$ consists of points. In this case the set $S_2(\mathcal{B})$ is empty, and we get the following corollary :

Corollary 1 *Let $\mathcal{B}$ be a set of points and $\mathcal{C}$ be the set of Voronoi circles of $\mathcal{V}(\mathcal{B})$. Then $\mathcal{V}(\mathcal{C})$ is a planar graph isomorphic to the Delaunay triangulation of the point set $\mathcal{B}$.*

The following lemma describes the geometry of the edges of $\mathcal{D}(\mathcal{B})$.

Lemma 8 *Let e be an edge of $\mathcal{D}(\mathcal{B})$. Then e is one of the following :*

1. *A line segment connecting the centers of two sites in $\mathcal{B}$.*
2. *A hyperbolic segment connecting the centers of two sites in $\mathcal{B}$.*
3. *A parabolic segment connecting the centers of two sites in $\mathcal{B}$.*
4. *A ray originating from the a center of a site in $\mathcal{B}$.*

Proof. Let e_{ij}^{kl} be an edge of $\mathcal{D}(\mathcal{B})$ and let α_{ij}^{kl} be the dual edge of e_{ij}^{kl} in $\mathcal{V}(\mathcal{B})$. Finally, we denote the site at infinity as B_∞. Let also R_{ijk} and R_{ijl} denote the weights of the Voronoi circles C_{ijk} and C_{ijl}, the centers of which are the two endpoints of α_{ij}^{kl}.

Consider the following cases, where we assume, without loss of generality that $r_i \leq r_j$ and that $R_{ijk} \leq R_{ijl}$:

1. $B_i, B_j \neq B_\infty$ and $R_{ijk} = R_{ijl}$. Then the bisector of C_{ijk} and C_{ijl} is a line and thus e_{ij}^{kl} is a line segment connecting b_i and b_j. Note that this holds true even if C_{ijk} and C_{ijl} are circles that go through B_∞, since then C_{ijk} and C_{ijl} are lines and the bisector of two lines is also a line.

2. $B_i, B_j \neq B_\infty$ and $R_{ijk} \neq R_{ijl} < \infty$. Then the bisector of C_{ijk} and C_{ijl} is a hyperbola and in this case e_{ij}^{kl} is section of hyperbola connecting b_i and b_j.

3. $B_i, B_j \neq B_\infty$ and $R_{ijk} < \infty$, $R_{ijl} = \infty$. In this case C_{ijl} is a circle that passes from the site at infinity B_∞. On the Euclidean plane C_{ijl} is a line tangent to B_i and B_j, and the bisector of C_{ijk} and C_{ijl} is the locus of points equidistant from a circle and a line, which is a parabola. Hence in this case e_{ij}^{kl} is a section of parabola delimited by b_i and b_j.

4. $B_i \neq B_\infty$ and $B_j \equiv B_\infty$. In this case both C_{ijk} and C_{ijl} are circles that go through B_∞, i.e., they are lines tangent to B_i. Their bisector is a also line. Since one of the endpoints of e is the point at infinity we have that in this case e_{ij}^{kl} is a ray originating from b_i.

□

Remark: The results of Lemma 8 are not restricted to the dual of the AW-Voronoi diagram of a set of circles. In fact, Lemma 8 describes the geometry of the AW-Voronoi diagram of a set of sites, where the sites can either be circles, with positive or negative weights, or half-planes.

Contents

1 Introduction **3**

2 Preliminaries **5**

3 Inserting a site incrementally **9**
3.1 Triviality test 9
3.2 Finding the conflict region 10
3.3 Storing the trivial sites 14
3.4 Runtime analysis 14

4 Nearest neighbor location **15**

5 Deleting a site **17**
5.1 Deleting a non-trivial site 17
5.2 Deleting a trivial site 18
5.3 Runtime analysis 18

6 Predicates and implementation **19**
6.1 Predicates 19
6.2 Implementation 19
6.3 Benchmarks 19

7 Conclusion **20**

A The dual graph **23**

Unité de recherche INRIA Sophia Antipolis
2004, route des Lucioles - BP 93 - 06902 Sophia Antipolis Cedex (France)

Unité de recherche INRIA Lorraine : LORIA, Technopôle de Nancy-Brabois - Campus scientifi que
615, rue du Jardin Botanique - BP 101 - 54602 Villers-lès-Nancy Cedex (France)
Unité de recherche INRIA Rennes : IRISA, Campus universitaire de Beaulieu - 35042 Rennes Cedex (France)
Unité de recherche INRIA Rhône-Alpes : 655, avenue de l'Europe - 38330 Montbonnot-St-Martin (France)
Unité de recherche INRIA Rocquencourt : Domaine de Voluceau - Rocquencourt - BP 105 - 78153 Le Chesnay Cedex (France)

Éditeur
INRIA - Domaine de Voluceau - Rocquencourt, BP 105 - 78153 Le Chesnay Cedex (France)
http://www.inria.fr
ISSN 0249-6399