The predicates of the Apollonius diagram: algorithmic analysis and implementation

Ioannis Z. Emiris [a]

[a] *Department of Informatics & Telecommunications, National Kapodistrian University of Athens, Panepistimiopolis 15784, Athens, Greece;* `emiris@di.uoa.gr`.

Menelaos I. Karavelas [b]

[b] *Department of Computer Science & Engineering, University of Notre Dame, Notre Dame, IN 46556, U.S.A.;* `mkaravel@cse.nd.edu`.

Abstract

We study the predicates involved in an efficient dynamic algorithm for computing the Apollonius diagram in the plane, also known as the additively weighted Voronoi diagram. We present a complete algorithmic analysis of these predicates, some of which are reduced to simpler and more easily computed primitives. This gives rise to an exact and efficient implementation of the algorithm, that handles all special cases. Among our tools we distinguish an inversion transformation and an infinitesimal perturbation for handling degeneracies.

The implementation of the predicates requires certain algebraic operations. In studying the latter, we aim at minimizing the algebraic degree of the predicates and the number of arithmetic operations; this twofold optimization corresponds to reducing bit complexity. The proposed algorithms are based on static Sturm sequences. Multivariate resultants provide a deeper understanding of the predicates and are compared against our methods. We expect that our algebraic techniques are sufficiently powerful and general to be applied to a number of analogous geometric problems on curved objects. Their efficiency, and that of the overall implementation, are illustrated by a series of numerical experiments. Our approach can be immediately extended to the incremental construction of abstract Voronoi diagrams for various classes of objects.

Key words: computational geometry, algebraic computing, Apollonius diagram, geometric predicates, Sturm sequence, resultant
2000 MSC: 68U05, 68W30

1 Introduction

Voronoi diagrams are among the most studied constructions in computational geometry due to their numerous applications, including motion planning and collision detection, communication networks, graphics, and growth of micro-organisms in biology.

In this paper we deal with the problem of computing the predicates for the *Apollonius diagram*, also known as the additively weighted Voronoi diagram. The input data is a set of points and a set of weights associated with them. We denote the Euclidean norm by $\|\cdot\|$. 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

$$\delta(p, B) = \|p - b\| - r.$$

We also define the distance $\delta(B_i, B_j)$ between two weighted points $B_\nu = \{b_\nu, r_\nu\}$, $\nu = i, j$, as:

$$\delta(B_i, B_j) = \|b_i - b_j\| - r_i - r_j.$$

The *Apollonius diagram* is then defined to be the subdivision of the plane induced by assigning each point $p \in \mathbb{E}^2$ to its nearest neighbor with respect to the distance function $\delta(\cdot, \cdot)$. If the weights are positive, the Apollonius diagram can be viewed as the Voronoi diagram for a set of circles. Points outside a circle have positive distance, whereas points inside a circle have negative distance. The Apollonius diagram does not change if all the weights are translated by the same quantity. Hence, in the remainder of this paper we assume that all the weights are positive. We will also use the term *site* to refer to a weighted point taken from our input set. In contrast to the usual Euclidean Voronoi diagram for points, in Apollonius diagram a site can have an empty Voronoi cell. We call such a site *hidden*.

There have been several algorithms for this problem, e.g. [1,2,3,4,5], however the problem of designing algorithms for the evaluation of predicates has seldom been treated and even less often implemented. In particular, [5,3] discuss the predicates required, but they are rather complicated. The algorithm presented in [4] treats Voronoi diagrams in an abstract way and thus requires the predicates as input. The work [6] examines the main predicates involved in the algorithm of [1], which is nonetheless of quadratic complexity and off-line. The algebraic formulations of the predicates have maximum degree 16 in the input variables, which is the same as in our case.

In [7,8], an implementation of the Delaunay triangulation of the input point set is used, followed by edge flips in order to arrive at the desired Voronoi diagram. However, the algorithm has quadratic worst-time complexity and is off-line; it uses a somewhat different metric, so all sites correspond to non-empty Voronoi cells. Lastly, this approach maintains topological consistency but not geometric exactness, and it makes no algebraic analysis of the predicates.

The combinatorial algorithm that we consider is a dynamic one and is detailed in [9]; the latter also offers a proof of correctness. The basic idea is similar to that in [4]. To insert a new site we first determine if the new site is hidden. Otherwise, we find the portion of the existing Apollonius diagram that is in conflict with the new site; finally we add the new site to the existing Apollonius diagram using the boundary of the conflict region. The predicates needed for this algorithm are discussed in [9, Sec.6]; we organize them in a flowchart in Section 3.3. It is noteworthy that a subset of our predicates are sufficient for implementing the algorithm in [4] for the same problem. In [4], the only predicate needed by the algorithm is the following: given an edge of the Voronoi diagram, defined by four sites, as well as a fifth site, determine what portion of the Voronoi edge is destroyed by the new site. We discuss this in more detail in Section 11.

The study of predicates in computational geometry indicates a shift of focus towards lower level algorithmic issues. In particular, minimizing the algebraic degree of the tested quantities (in terms of the input parameters) has nowadays become a question that influences algorithm design. A related issue concerns algebraic algorithms for evaluating geometric predicates efficiently and accurately, e.g. [10,11,12,13,14]. In Section 8.5 we deal with one such predicate, which calls for comparing roots of real quadratic polynomials; our algebraic techniques have been published in preliminary form in [15] and achieve a maximum degree of the tested quantities equal to 16. The study of several other possible methods has been undertaken in [16]. Its main conclusion is that all methods which minimize the maximum algebraic degree need to test the same quantities. We aspire that our algebraic tools, based on static Sturm sequences, can serve not only for solving the problem in higher dimensions but also in analogous problems, such as arrangements of algebraic curves and surfaces. Their advantage is that they do not depend on the root separation. Moreover, all computations can be performed over the ring of the given quantities thus, if all inputs are integers, it suffices to use integer arithmetic of sufficiently high precision. On the downside, our algebraic methods rely on isolating intervals for the roots of polynomials, in order to guarantee optimal performance. Extensions of our approach to polynomials of degree higher than 2 can be found in [17].

This paper is structured as follows. In the following section we discuss the problem's setting and properties of the Apollonius diagram. In Section 3 we sketch the dynamic algorithm and illustrate the relations among its predicates and subpredicates. Our main algebraic tools are discussed in Section 4.

In Section 4.3 we describe inversion, and perform some preliminary computations that will be needed for the (sub)predicates. In Section 5 we present our perturbation scheme for dealing with degeneracies. Section 6 treats the first three predicates, while Section 7 presents the InfiniteEdgeConflictType predicate. The next predicate is analyzed in Section 8, along with its subpredicates: Section 8.1 shows how to determine the type of a shadow region and the next two sections describe the Existence and the InCircle subpredicates. In Section 8.5 we describe the RadiiDifference subpredicate and show how to compute it optimally; this is the hardest predicate and our algebraic tools are fully applied here. Section 9 treats the last two predicates. In Section 10 we perform two sets of numerical experiments, namely for studying the entire algorithm and its predicates. Finally, in Section 11 we discuss extensions of our work and we conclude with open questions and future work.

2 Preliminaries

In this section we provide basic definitions and discuss various properties of Apollonius diagrams that are interesting from the algorithmic or evaluation-of-predicates point of view. When applicable we will make the analogy or indicate the differences with respect to the Voronoi diagram of point sites.

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) Apollonius 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 Apollonius cells are called *Apollonius edges*, whereas points that belong to more than two Apollonius cells are called *Apollonius vertices*. The *Apollonius diagram* $\mathcal{V}(\mathcal{B})$ of $\mathcal{B}$ is defined as the collection of the Apollonius cells, edges and vertices. The *Apollonius skeleton* $\mathcal{V}_1(\mathcal{B})$ of $\mathcal{B}$ is defined as the union of the Apollonius edges and Apollonius vertices of $\mathcal{V}(\mathcal{B})$. The Apollonius diagram is a subdivision of the plane [5, Property 1]. Its skeleton consists of straight or hyperbolic arcs and each cell is star-shaped with respect to the center of the corresponding site [5, Properties 3 and 4]). This is entirely analogous to the case of points, with the only exception being that the skeleton consists of straight arcs only.

In the case of the usual Euclidean Voronoi diagram for a set of points, every point has a non-empty Voronoi cell. In Apollonius diagrams there may exist sites, the Apollonius cells of which are empty. In particular, the Apollonius cell V_i of a site B_i is empty if and only if B_i is contained in another site B_j (see [5, Property 2]). A site whose Apollonius cell has empty interior is called *hidden*, whereas a site whose Apollonius cell has non-empty interior is called

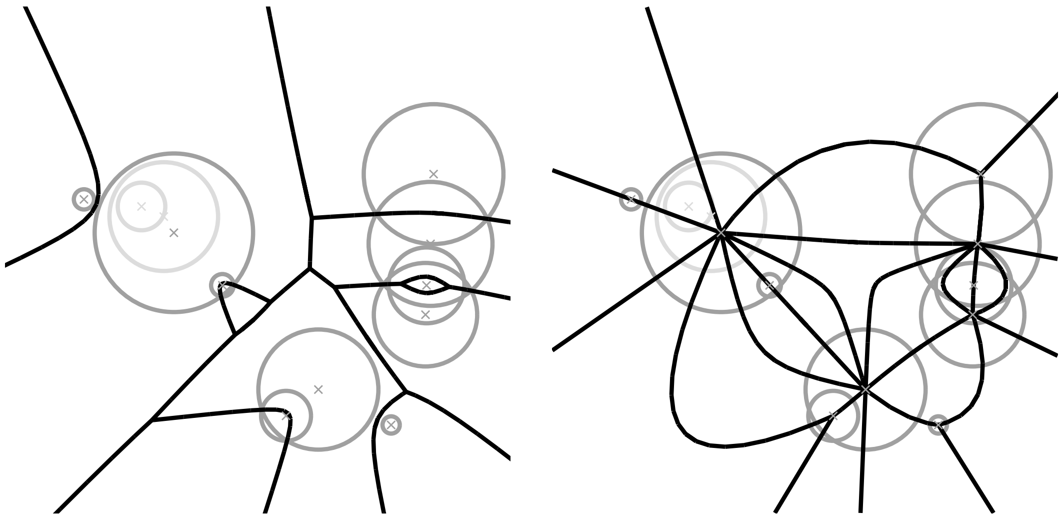


Fig. 1. Left: the Apollonius diagram for a set of 12 sites. Visible sites are shown in dark gray. Hidden sites are shown in light gray. The Apollonius skeleton is shown in black. Right: a planar embedding of the Apollonius graph of the same set of sites. The edges of the Apollonius graph are shown in black.

visible. Fig. 1(left) shows the Apollonius diagram for a set of 12 sites, among which 2 are hidden.

We call Apollonius graph and note $\mathcal{D}(\mathcal{B})$ the dual graph of the Apollonius diagram $\mathcal{V}(\mathcal{B})$. There is a vertex in $\mathcal{D}(\mathcal{B})$ for each visible site B_i in $\mathcal{B}$. Let B_i and B_j be two sites whose Apollonius cells V_i and V_j are adjacent. We denote by $\alpha_{ij}^{k\ell}$ the Apollonius edge in $V_i \cap V_j$ whose endpoints are the Apollonius vertices equidistant to B_i, B_j, B_k and B_i, B_ℓ, B_j, respectively. There exists an edge $e_{ij}^{k\ell}$ in $\mathcal{D}(\mathcal{B})$ connecting B_i and B_j for each edge $\alpha_{ij}^{k\ell}$ of $\mathcal{V}(\mathcal{B})$ in $V_i \cap V_j$. The fact that we have a planar embedding of linear size for the Apollonius graph [5, Property 7], immediately implies that the size of the Apollonius diagram is $O(n)$. The Apollonius skeleton may consist of more than one connected component [5, Property 9], whereas the dual graph is always connected.

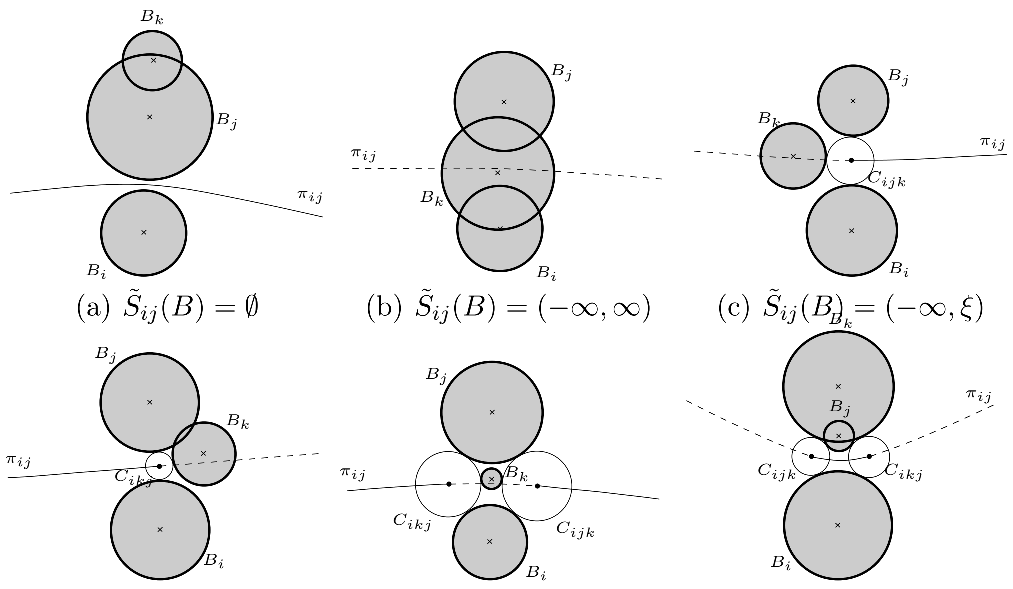
If we do not have any degeneracies, the Apollonius graph has the property that all but its outer face have exactly three edges, which is exactly what happens in the case of the Delaunay triangulation for points. Unlike Delaunay triangulations, Apollonius graphs may contain vertices of degree 2, i.e., we have triangular faces with two edges in common. Moreover, if the Apollonius skeleton consists of more than one connected component, the Apollonius graph may also have vertices of degree 1, which are the dual of Apollonius edges with no vertices (e.g., the Apollonius edge at the top left corner of Fig. 1(left)). To simplify the representation of the Apollonius graph we add a fictitious site called the site at infinity B_∞. This amounts to adding a Apollonius 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 Apollonius edges forming the boundary of the Apollonius cell of B_∞. In this compactified version, the Apollonius skeleton consists of only one connected component, and the previously non-connected components are now connected through the edges of the Apollonius cell of B_∞. The compactified Apollonius graph corresponds to the original Apollonius graph plus edges connecting the sites on the convex hull of $\mathcal{B}$ with B_∞. In the absence of degeneracies, all faces of the compactified Apollonius graph have exactly three edges, but this graph may still have vertices of degree 2. From now on when we refer to the Apollonius diagram or the Apollonius graph, we refer to their compactified versions (see Fig. 1(right)). Note that in the case of point sites, both the original and the compactified versions of Voronoi diagrams consist of a single connected component, whereas in Delaunay triangulations no edges of degree 2 ever appear.

Degenerate cases arise when there are points equidistant to more than three sites. Then, the Apollonius 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 Apollonius graph through an arbitrary "triangulation" of the faces with more than three edges.

Let B_i and B_j be two sites such that none is contained inside the other. Let us, moreover, assume that neither B_i nor B_j is the site at infinity B_∞. 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 Apollonius circle*. A circle tangent to B_i and B_j that lies in $B_i \cap B_j$ is an *interior bitangent Apollonius circle*. Let B_i, B_j and B_k be three sites, such that none 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 Apollonius 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 Apollonius circle*. A triple of sites B_i, B_j and B_k can have up to two tritangent Apollonius circles, either exterior or interior. This is equivalent to stating that the Apollonius diagram of three sites can have up to two Apollonius vertices (see [5, 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 Apollonius circles. Let also $\mathrm{CCW}(\cdot,\cdot,\cdot)$ denote the usual *orientation test* of three points. If $\mathrm{CCW}(p_i, p_j, p_k) > 0$ we say that the tritangent Apollonius circle is a CCW-*Apollonius circle* of the triple B_i, B_j, B_k. If $\mathrm{CCW}(p_i, p_j, p_k) < 0$, we say that the tritangent Apollonius circle is a CW-*Apollonius circle* of the triple B_i, B_j, B_k. We will show in Section 8.2 that three sites in a given order can have at most one CCW- or CW-Apollonius circle, which can be either exterior or interior.

The situation here is drastically different with respect to the points' case. In the points' case there is really no distinction between interior and exterior



(a) $\tilde{S}_{ij}(B) = \emptyset$ (b) $\tilde{S}_{ij}(B) = (-\infty, \infty)$ (c) $\tilde{S}_{ij}(B) = (-\infty, \xi)$

(d) $\tilde{S}_{ij}(B) = (\eta, \infty)$ (e) $\tilde{S}_{ij}(B) = (\xi, \eta)$ (f) $\tilde{S}_{ij}(B) = (-\infty, \xi) \cup (\eta, \infty)$

Fig. 2. The 6 possible cases for the shadow region $S_{ij}(B_k)$ (dashed curve(s)). The Apollonius circles C_{ijk} and C_{ikj} are shown only if they exist (thin white circles).

tritangent circles – in fact interior tritangent circles can never exist. Viewing points as special cases of circles, a circle passing through three points is nothing but a *double* circle (in the algebraic sense), i.e., two copies of the same circle. The fact, however, that we have a single CCW-Apollonius circle is the analog of the well known fact that there exists a unique circle that passes through three (non-collinear) points.

Let π_{ij} denote the bisector of the sites B_i and B_j. As we have 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_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 $\mathbb{R}$. Given a bitangent Apollonius 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 Apollonius 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 $\emptyset$, $(-\infty, \infty)$, $(-\infty, \xi)$, (η, ∞), (ξ, η) and $(-\infty, \xi) \cup (\eta, \infty)$, where $\xi, \eta \in \mathbb{R}$ (see Fig. 2).

Let $\alpha_{ij}^{k\ell}$ be an edge of $\mathcal{V}(\mathcal{B})$, and let C_{ijk} and $C_{i\ell j}$ be the tritangent CCW-

Apollonius circles associated with the endpoints of $\alpha_{ij}^{k\ell}$. We denote by c_{ijk} (resp. $c_{i\ell j}$) the center of C_{ijk} (resp. $C_{i\ell j}$) and call c_{ijk} (resp. $c_{i\ell j}$) the ijk-endpoint or ijk-vertex (resp. $i\ell j$-endpoint or $i\ell j$-vertex) of $\alpha_{ij}^{k\ell}$. Under the mapping $\zeta_{ij}(\cdot)$, $\alpha_{ij}^{k\ell}$ maps to the interval $\tilde{\alpha}_{ij}^{k\ell} = [\xi_{ijk}, \xi_{i\ell j}] \subset \mathbb{R}$ (if $k = \infty$ or $\ell = \infty$, then $\tilde{\alpha}_{ij}^{k\ell} = (-\infty, \xi_{i\ell j}]$ or $\tilde{\alpha}_{ij}^{k\ell} = [\xi_{ijk}, \infty)$, respectively). We define the *conflict region* $R_{ij}^{k\ell}(B)$ of B with respect to the edge $\alpha_{ij}^{k\ell}$ to be the intersection $R_{ij}^{k\ell}(B) = \alpha_{ij}^{k\ell} \cap S_{ij}(B)$. Then, B *is in conflict* with $\alpha_{ij}^{k\ell}$ if $R_{ij}^{k\ell}(B) \neq \emptyset$. Under the mapping by $\zeta_{ij}(\cdot)$, the conflict region $R_{ij}^{k\ell}(B)$ maps to $\tilde{R}_{ij}^{kl\ell}(B) = \tilde{\alpha}_{ij}^{k\ell} \cap \tilde{S}_{ij}(B)$. $\tilde{R}_{ij}^{k\ell}(B)$ can be one of the following types (see also Fig. 3):

(1) $\tilde{R}_{ij}^{k\ell}(B) = \emptyset$, in which case we say that B *is not in conflict* with $\alpha_{ij}^{k\ell}$.

(2) $\tilde{R}_{ij}^{k\ell}(B)$ consists of a single connected interval, in which case we further distinguish between the following cases:
   (a) $\tilde{\alpha}_{ij}^{k\ell} = \tilde{R}_{ij}^{k\ell}(B)$, in which case we say that B *is in conflict with the entire edge* $\alpha_{ij}^{k\ell}$.
   (b) $\tilde{R}_{ij}^{k\ell}(B)$ contains ξ_{ijk}, but not $\xi_{i\ell j}$, in which case we say that B *is in conflict with the* ijk*-vertex (or first vertex) of* $\alpha_{ij}^{k\ell}$.
   (c) $\tilde{R}_{ij}^{k\ell}(B)$ contains $\xi_{i\ell j}$, but not ξ_{ijk}, in which case we say that B *is in conflict with the* $i\ell j$*-vertex (or second vertex) of* $\alpha_{ij}^{k\ell}$.
   (d) $\tilde{R}_{ij}^{k\ell}(B)$ contains neither ξ_{ijk} nor $\xi_{i\ell j}$, in which case we say that B *is in conflict with the interior of* $\alpha_{ij}^{k\ell}$.

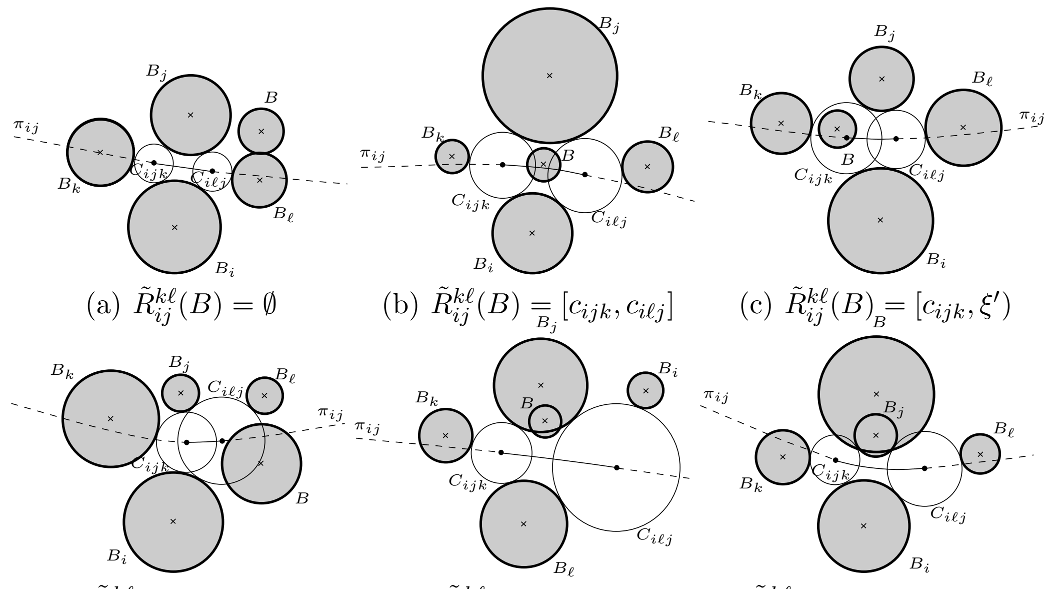
(3) $\tilde{R}_{ij}^{k\ell}(B)$ consists of two disjoint intervals, including respectively ξ_{ijk} and $\xi_{i\ell j}$, in which case we say that B *is in conflict with both vertices of* $\alpha_{ij}^{k\ell}$.

Finally we define the *conflict region* $R_{\mathcal{B}}(B)$ of B with respect to $\mathcal{B}$ as the union

$$R_{\mathcal{B}}(B) = \bigcup_{\alpha_{ij}^{k\ell} \in \mathcal{V}(\mathcal{B})} R_{ij}^{k\ell}(B).$$

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 Apollonius cell of B in $\mathcal{V}(\mathcal{B} \cup \{B\})$.

Consider now the case where either B_i or B_j is the site at infinity. Without loss of generality we assume that $B_j \equiv B_\infty$. In this case the bitangent Apollonius circles correspond to lines tangent to B_i and they are always exterior. The CCW-tritangent Apollonius circle $C_{i\infty k}$ of B_i, B_∞ and B_k becomes an oriented line bitangent to B_i and B_k, that has both B_i and B_k to its left and touches B_i, B_k in this order. The bisector $\pi_{i\infty}$ is now a bisector at infinity, but we can still define the map $\zeta_{i\infty}(\cdot)$ as follows. Let p be a point on $\pi_{i\infty}$ and let $W_{i\infty}(p)$ be the corresponding bitangent Apollonius circle, i.e., in this case, an oriented line tangent to B_i at some point t, that has B_i to its left. Let $\vec{\nu}_{i\infty}(p)$ denote the unit vector in the direction of tb_i. $\vec{\nu}_{i\infty}(p)$ is perpendicular to $W_{i\infty}(p)$ and defines a (unique) point on the unit circle $\mathbb{S}^1$. We define $\vec{n}_{i\infty}(p)$ to be the image of p through $\zeta_{i\infty}(\cdot)$. Hence, the function $\zeta_{i\infty}(\cdot)$ is 1–1 and onto mapping from $\pi_{i\infty}$ to $\mathbb{S}^1$. An Apollonius edge $\alpha_{i\infty}^{k\ell}$ on $\pi_{i\infty}$ maps to an oriented circular



(a) $\tilde{R}_{ij}^{k\ell}(B) = \emptyset$ (b) $\tilde{R}_{ij}^{k\ell}(B) = [c_{ijk}, c_{i\ell j}]$ (c) $\tilde{R}_{ij}^{k\ell}(B) = [c_{ijk}, \xi')$

(d) $\tilde{R}_{ij}^{k\ell}(B) = (\eta', c_{i\ell j}]$ (e) $\tilde{R}_{ij}^{k\ell}(B) = (\xi', \eta')$ (f) $\tilde{R}_{ij}^{k\ell}(B) = [c_{ijk}, \xi') \cup (\eta', c_{i\ell j}]$

Fig. 3. The 6 possible cases for the conflict region $R_{ij}^{k\ell}(B)$ of an Apollonius edge $\alpha_{ij}^{k\ell}$ lying on a finite bisector. The edge $\alpha_{ij}^{k\ell}$ is shown as a solid thin curve.

arc on $\mathbb{S}^1$ and similarly the shadow region $S_{i\infty}(B_k)$ of B_k with respect to $\pi_{i\infty}$ maps also to an oriented circular arc on $\mathbb{S}^1$. In order to deduce the type of the conflict region $R_{i\infty}^{k\ell}(B)$ it suffices to determine the type of the intersection of the two circular arcs $\tilde{\alpha}_{i\infty}^{k\ell}$ and $\tilde{S}_{i\infty}(B)$. The types of the conflict region are exactly the same as those in the case of a finite bisector. In the subsequent sections we will also identify $\mathbb{S}^1$ with B_i, since every point $\vec{n}_{i\infty}(\cdot)$ on $\mathbb{S}^1$ is in 1–1 correspondence with the point $b_i + \vec{n}_{i\infty}(\cdot)r_i$ on B_i.

Let us finish this section by again making the comparison against the case of point sites. In the point Voronoi diagram the notions of shadow and conflict region are entirely analogous. What is different are the possible cases. The shadow region of a point p with respect to the bisector π_{ij} of two other points p_i and p_j can only be one of the following four types: $\emptyset$ (where p is collinear with p_i and p_j but does not belong to the segment p_ip_j), $(-\infty, \infty)$ (when p is contained in the segment p_ip_j), $(-\infty, \xi)$ (when p is to the left of the line passing through p_i and p_j) and (η, ∞) (when p is to the right of the line passing through p_i and p_j). The possible cases for the conflict region of a point with respect to a Voronoi edge are also four, namely the cases 1, 2(a), 2(b) and 2(c) above. Another way to derive this is consider the relation $\tilde{R}_{ij}^{k\ell}(p) = \tilde{\alpha}_{ij}^{k\ell} \cap \tilde{S}_{ij}(p)$: here $\tilde{S}_{ij}(p)$ is any of the four types mentioned above and $\tilde{\alpha}_{ij}^{k\ell}$ is an interval. The possible types for the intersection $\tilde{R}_{ij}^{k\ell}(p)$ are also four and topologically equivalent to the types for $\tilde{S}_{ij}(p)$.

3 The dynamic algorithm

In this section we describe briefly the algorithm presented in [9]. The aim of this section is to focus on the required predicates rather than explain in detail the combinatorial aspects of the algorithm. The interested reader should refer to [9] for the details of the combinatorial part of the algorithm.

3.1 Inserting a site incrementally

In this subsection we show how to insert a new site. Let $\mathcal{B}$ be our set of n sites and let us assume that we have already constructed the Apollonius diagram for a subset $\mathcal{B}'$ of $\mathcal{B}$. We now want to insert a site $B \notin \mathcal{B}'$. The insertion is done in the following steps:

(1) Locate the nearest neighbor $NN(B)$ of B in $\mathcal{B}'$, with respect to the distance function $\delta(B, \cdot)$.
(2) Test if B is hidden.
(3) Find the conflict region of B and repair the Apollonius graph.

3.1.1 Nearest neighbor location.

The nearest neighbor location of B in fact reduces to the location of the center b of B in $\mathcal{V}(\mathcal{B}')$. We can do that as follows. Select a site $B_i \in \mathcal{B}'$ at random. Look at all the neighbors of B_i in the Apollonius graph. If there exists a B_j, $j \neq i$, such that $\delta(B, B_j) < \delta(B, B_i)$, then B_i cannot be the nearest neighbor of B. In this case we replace B_i by B_j and restart our procedure. If none of the neighbors of B_i is closer to B than B_i, then $NN(B) = B_i$.

Clearly, the only predicate needed for this phase of the algorithm is to compare the quantities $\delta(B, B_j)$ and $\delta(B, B_i)$. In fact this comparison reduces to comparing the quantities $\delta(b, B_j)$ and $\delta(b, B_i)$. Geometrically, with this predicate we determine the half-plane, with respect to the (oriented) bisector π_{ij} of B_i and B_j, which contains the point b. We refer to this predicate as the SideOfBisector predicate.

3.1.2 Testing if a site is hidden.

It is shown in [9, Lemma 1] that B is hidden if and only if $B \subset NN(B)$. This amounts to determining the sign of the quantity $\delta(B, NN(B)) + 2r$, where r is the radius of B. The required predicate determines, given a site B_i and a query site B, whether $B \subset B_i$. We call this predicate IsHidden.

3.1.3 Finding the conflict region.

Let $R'(B)$ be the conflict region of B with respect to $\mathcal{B}'$. Let $\partial R'(B)$ denote the boundary of $R'(B)$. As discussed in [9], in order to insert the new site B we need to discover $\partial R'(B)$. This is done as follows. First we find a point on the Apollonius skeleton $\mathcal{V}_1(\mathcal{B})$, that is in conflict with B. Starting at that point we perform a *depth first search* (DFS) on the Apollonius skeleton to discover the entire region $R'(B)$. Once we have discovered $R'(B)$, we also have $\partial R'(B)$.

We consider finding a first point on the Apollonius skeleton $\mathcal{V}_1(\mathcal{B}')$ which is in conflict with B. By [9, Lemma 2], if B is visible it has to be in conflict with at least one of the Apollonius edges of the Apollonius cell $V_{NN(B)}$ of its nearest neighbor $NN(B)$ in $\mathcal{B}'$. Since $V_{NN(B)}$ is star-shaped, the Apollonius vertices of $V_{NN(B)}$ split $V_{NN(B)}$ into portions of cones whose apex is the center $b_{NN(B)}$ of $NN(B)$. Clearly, we can associate each Apollonius edge of $V_{NN(B)}$ with such a cone. Then B is in conflict with the edge α, the cone of which contains the center b of B. Therefore, we need to locate the cone that contains b, which reduces to orientation tests of the form $\mathrm{CCW}(v, b, b_{NN(B)})$, where v is an Apollonius vertex of $V_{NN(B)}$.

The corresponding predicate is Orientation. Formally, the input for this predicate is a tritangent Apollonius circle C_{ijk} and two sites B_ℓ, B_m, and returns the result of the test $\mathrm{CCW}(c_{ijk}, b_\ell, b_m)$, where c_{ijk}, b_ℓ and b_m are the centers of C_{ijk}, B_ℓ and B_m, respectively.

Suppose that we have found a Apollonius edge α on the boundary of $V_{NN(B)}$ that is in conflict with B. If B is in conflict with the interior of α, we have discovered the entire conflict region $R'(B)$. Otherwise, B has to be in conflict with at least one of the two Voronoi vertices of α. In this case the DFS algorithm is invoked and will recursively visit all vertices in conflict with B. Suppose that we have arrived at a Apollonius vertex v (which is a node on the Apollonius skeleton). Firstly, we mark it. Then we look at all the Apollonius edges α adjacent to it. Let v' be the Apollonius vertex of α that is different from v. If v' has been marked then the DFS backtracks, since we have already processed v'. If v' has not been visited we determine the type of the conflict region of B with α. If B is in conflict with the entire edge α we continue recursively on v'. Otherwise, the DFS backtracks.

From the point of view of the predicates required for the DFS performed, the only operation we need is to find the conflict type of an Apollonius edge α given a site B. This constitutes the EdgeConflictType predicate.

3.2 Site deletion

During the insertion procedure hidden sites can appear in two possible ways. Either the new site B to be inserted is hidden, or B contains existing sites, which after the insertion of B will become hidden. When deletion of sites is allowed, B may contain other sites which will become visible if B is deleted. Since a site is hidden if and only if it is contained inside some other site, there exists a natural parent-child relationship between hidden and visible sites. In particular, we can associate every hidden site to a visible site that contains it. If a hidden site is contained in more than one visible sites, we can choose the parent of the hidden site arbitrarily. A natural choice for storing hidden sites is to maintain a list for every visible site, which contains all hidden sites that have the visible site as their parent. In the sequel of this subsection, we denote by $L_h(B)$ the list of sites whose parent is B.

Suppose we are given a set $\mathcal{B}$ of sites for which we have already constructed the Apollonius diagram $\mathcal{V}(\mathcal{B})$. Let also $B \in \mathcal{B}$ be a site that we wish to delete from $\mathcal{V}(\mathcal{B})$. We distinguish between the cases where B is visible or hidden.

3.2.1 Deleting a visible site.

Suppose that B is visible. Let $\mathcal{B}_\gamma$ be the set of neighbors of V_B in $\mathcal{D}(\mathcal{B})$. By [9, Lemma 6], the Apollonius diagram after the deletion of B can be found by constructing the Apollonius diagram of $\mathcal{B}_\gamma \cup L_h(B)$. Once $\mathcal{V}(\mathcal{B}_\gamma \cup L_h(B))$ has been constructed, we can construct $\mathcal{V}(\mathcal{B} \setminus \{B\})$ by superimposing the two Apollonius diagrams.

Recall that we represent the Apollonius diagram through its dual, the Apollonius graph. Given this, the above mentioned superposition of Apollonius diagrams is trivial if there are no degeneracies in the input, since in this case the dual graphs are locally the same. However, if degeneracies exist, it is possible that the dual graphs are not locally the same; this is due to the fact that they depend on the order of insertion of sites. We will describe in Section 5 how we deal with degeneracies.

The required predicate determines, given an Apollonius edge α, whether this edge is degenerate, i.e., it has empty interior. We call this predicate IsDegenerateEdge.

3.2.2 Deleting a hidden site.

Suppose that B is hidden. We have to find the visible site B_i such that $B \in L_h(B_i)$ and then delete B from $L_h(B_i)$. By [9, Lemma 1], $B \subset NN(B)$.

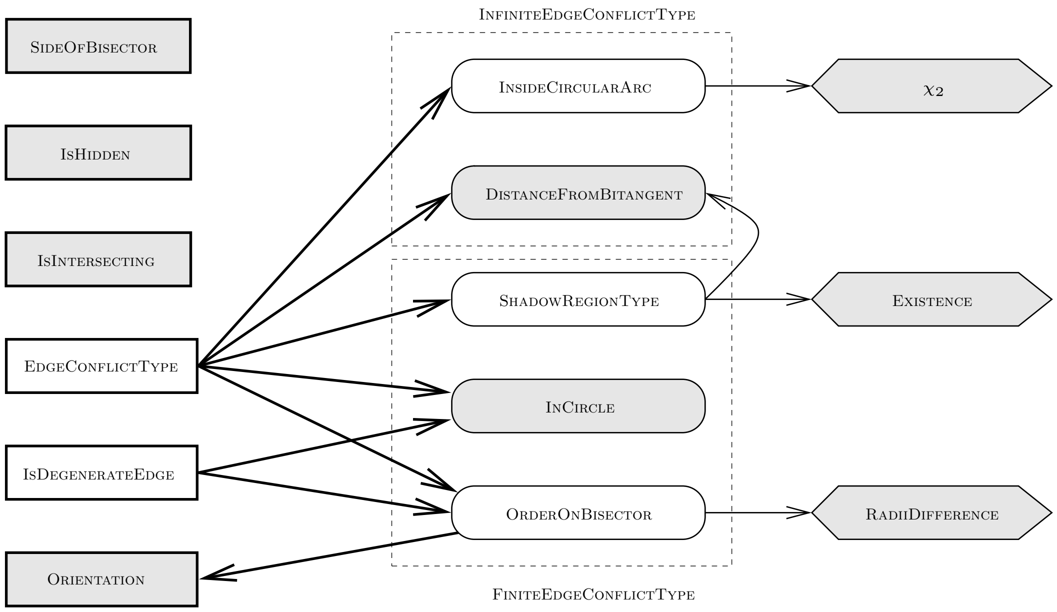


Fig. 4. Relation of predicates, subpredicates and primitives. Shaded boxes correspond to the lowest level operations; these are the operations that need to be implemented.

Hence B must be in the list $L_h(B')$ of some B_i, 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 [18, Chapter 5]. Hence the deletion of B can be done as follows:

(1) Find the nearest neighbor $NN(B)$ of B;
(2) Walk on the connected component $\mathcal{C}_{NN(B)}$ of $NN(B)$ in the graph $\mathcal{K}(\mathcal{B})$ and for every site $B_i \in \mathcal{C}_{NN(B)}$ that contains B, test if $B \in L_h(B_i)$;
(3) Once the site B_i, such that $B \in L_h(B_i)$, is found, delete B from $L_h(B_i)$.

The only new operation that is needed is to determine if two sites B_i, B_j intersect. The predicate is equivalent to determining the sign of the quantity $\delta(B_i, B_j)$ and is called IsIntersecting.

3.3 Algorithmic analysis of the predicates

The EdgeConflictType predicate requires certain subpredicates for its evaluation. The subpredicates required depend on whether the corresponding Apollonius edge α lies on an infinite or finite bisector. In this context we conceptually have two versions of the EdgeConflictType predicate, namely the InfiniteEdgeConflictType and FiniteEdgeConflictType predicates. However, we assume that the EdgeConflictType predicate can recognize whether α is on an infinite or finite bisector and appropriately calls the corresponding subpredicates. If the bisector is infinite we need the following subpredicates:

- DistanceFromBitangent, which computes the sign of the distance of a site from a bitangent line of two other sites. It is equivalent to the InCircle predicate on the input data before performing the inversion mapping (described below).
- InsideCircularArc, which given a circular arc on a circle and a query point on the same circle determines if the point is inside the circular arc or not. In our case, both the endpoints of the circular arc, as well as the query point are defined as points of tangency of a bitangent line of two circles. We show how to reduce this subpredicate to the primitive operation χ_2 discussed and analyzed in [19]. In short, $\chi_2(\vec{a}, \vec{b})$ returns the sign of the 2-by-2 determinant of coordinates of $\vec{a}$ and $\vec{b}$, where $\vec{a}$ and $\vec{b}$ are vectors perpendicular to bitangent lines of two circles.

If the bisector is finite we need the following subpredicates:

- InCircle, which decides if a tritangent Apollonius circle is in conflict with an input site; it reduces to the classical InCircle predicate for points when all input radii are equal.
- Determining the type of a shadow region, called the ShadowRegionType subpredicate. This subpredicate is decided by 2 primitive operations: the Existence primitive, and DistanceFromBitangent.
- Ordering two points on the (oriented) bisector of two input sites, called the OrderOnBisector subpredicate; typically, these points are the centers of tritangent Apollonius circles, i.e., defined by the two sites and one third site for each. The primitive operations required here are Orientation and RadiiDifference, which are examined in detail later. In short, RadiiDifference compares the difference of the weights of two Apollonius circles whose centers lie on the bisector of two sites.

Finally, the IsDegenerateEdge predicate can be evaluated easily using the InCircle and OrderOnBisector subpredicates. We will describe in detail how this can be done in Section 9.2.

Figure 4 shows the relationships between the various, predicates, subpredicates and primitives required. The shaded boxes correspond to lowest level operations that need to be implemented.

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 latter constitute the input variables, or parameters. Computing the algebraic degree requires that we bound the degree of each polynomial whose sign needs to be tested (for the particular input) in order for the corresponding predicate to be decided. These polynomials are defined over the input variables, hence their total algebraic degree in these variables is well-defined.

4 Algebraic and geometric tools

This section introduces our main algebraic and geometric tools in three subsections, namely resultants, Sturm sequences, and the inversion transformation.

4.1 Multivariate resultants

We start with multivariate resultants in general and give Example 1; then we apply resultants to the problems encountered in our algorithm's predicates. The interested reader may consult [20,21] for details on resultants. This subsection presents also geometric invariants, which are also useful in conjunction with Sturm sequences, and Descartes' rule of sign.

Consider a system of $n+1$ polynomials in n affine variables, whose coefficients are *indeterminate* parameters. The *resultant* R of this system is an irreducible polynomial in these indeterminate parameters, with integer coefficients; see, for instance, the matrix determinant in Example 1. The resultant is well-defined up to a sign. It is possible to specialize all indeterminate coefficients to values in some arbitrary field. Then, the resultant evaluates to zero if and only if the specialized polynomials have a common root in the algebraic closure of the coefficient field.

More precisely, the *projective (or classical) resultant* of the homogenized polynomials in $n+1$ variables vanishes exactly when there exists a common root in projective space $\mathbb{P}^n$. *Toric (or sparse) resultants* express the existence of roots in a toric variety, which is the closure of $(\mathbb{C} \setminus \{0\})^n$ in a projective space of dimension usually larger than n. The resultant has degree $\deg_{f_i} R$ in the coefficients of each f_i equal to the generic number of roots of the other n polynomials, in the corresponding variety, either projective or toric. In the former case, Bézout's number implies $\deg_{f_i} R = \prod_{j \neq i} \deg f_j$, where $\deg f_j$ is the total degree of f_j. Toric elimination theory generalizes this to the mixed volume of the f_j for $j \neq i$. Typically, we wish to express the R as a determinant. The type of matrices on which we concentrate here is named after Sylvester.

When we are given a well-constrained system $f_1, \ldots, f_n$, one may define an over-constrained system by adding an *extra polynomial* $f_0 = u_0 + u_1 x_1 + \cdots + u_n x_n$. This yields the u-resultant, which factors into a constant term and linear factors $(u_0\alpha_0 + u_1\alpha_1 + \cdots + u_n\alpha_n)^m$, where $(\alpha_0 : \alpha_1 : \cdots : \alpha_n)$ is a common projective zero of the $f_1, \ldots, f_n$ and m its multiplicity.

Example 1 *Let us consider the bivariate system*

$$f_0 = u_0 + u_1x_1 + u_2x_2, f_1 = c_{10} + c_{11}x_1 + c_{12}x_1^2, f_2 = c_{20} + c_{21}x_2 + c_{22}x_2^2.$$

Clearly, the coefficients are indeterminate parameters, so the resultant shall be a polynomial in these u_i's and c_{ij}'s. A Sylvester-type matrix expressing the resultant is

$$M = \begin{bmatrix} c_{10} & c_{12} & c_{11} & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & c_{10} & c_{11} & c_{12} & 0 \\ c_{20} & 0 & 0 & c_{22} & c_{21} & 0 & 0 & 0 \\ 0 & 0 & c_{20} & 0 & 0 & c_{21} & 0 & c_{22} \\ u_0 & 0 & u_1 & 0 & u_2 & 0 & 0 & 0 \\ 0 & u_1 & u_0 & 0 & 0 & u_2 & 0 & 0 \\ 0 & 0 & 0 & u_2 & u_0 & u_1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & u_0 & u_1 & u_2 \end{bmatrix},$$

where the matrix columns are indexed by monomials

$$1, x_1^2, x_1, x_2^2, x_2, x_1x_2, x_1^2x_2, x_1x_2^2.$$

The rows of M express multiples of f_1, f_2, f_0 by the monomials $\{1, x_2\}$, $\{1, x_1\}$ and $\{1, x_1, x_2, x_1x_2\}$, respectively. The number of rows containing each of f_0, f_1, f_2 gives the degree of the determinant in the corresponding coefficients, namely 4,2 and 2. The Bézout numbers, as well as the mixed volumes, are 4,2,2 so, in this example, the toric and projective resultants coincide. It is easy to prove that $R \mid \det M$, hence $R = \det M$.

The rest of this section applies resultants in conjunction with invariants to a general class of predicates, which shall include RadiiDifference. It must be observed that the algebraic problem is similar to that in [10], concerning circular arcs. This is not obvious here, but will be demonstrated later. Consider predicates formulated by polynomials $f_\tau(x_\tau) := \alpha_\tau x_\tau^2 - 2\beta_\tau x_\tau + \gamma_\tau$, $\tau = 1, 2$, and $f_0 = -t + x_1 - x_2$. The latter is a u-polynomial when $u_0 \mapsto -t, u_1 \mapsto 1, u_2 \mapsto -1$. The resultant is of degree 4 in t and factors into a constant term and 4 linear terms $t - x_1 + x_2$, where the x_1, x_2 represent the 4 different pairs of roots. The irreducibility of the resultant for generic coefficients $\alpha_\tau, \beta_\tau, \gamma_\tau$ is tantamount to the fact that it gives us the smallest condition for the solvability of the 3 bivariate polynomials. Hence we obtain a lower bound on the quantities whose sign must be determined in order to decide the predicate.

The resultant coefficients can be simplified in terms of the classical geometric invariants, just as in [10]. For a comprehensive treatment of invariant theory see [22]. The quantities needed here are:

$$\Delta_\tau = \beta_\tau^2 - \alpha_\tau\gamma_\tau,\ \tau = 1, 2,\ K = \alpha_1\gamma_2 + \alpha_2\gamma_1 - 2\beta_1\beta_2.$$
$$J = \alpha_1\beta_2 - \alpha_2\beta_1,\ J' = \beta_1\gamma_2 - \beta_2\gamma_1,\ G = \alpha_1\gamma_2 - \alpha_2\gamma_1.$$

The Δ_τ's and K are the classical invariants for the f_τ by the action of $\mathrm{SL}(\mathbb{C}^2)$. This is the group of matrices whose determinant equals 1, a subgroup of $\mathrm{GL}(\mathbb{C}^2)$, the group of 2×2 invertible complex matrices. Moreover, J and J' are invariant with respect to translations for the system of two quadratic equations f_τ. G is not an invariant but its expression looks like one. We can now write,

$$R(t) = (\alpha_1\alpha_2)^2 t^4 + 4\alpha_1\alpha_2 J t^3 + (4J^2 + 2\alpha_1\alpha_2 K)t^2 + 4KJt + (G^2 - 4JJ'). \quad (1)$$

For a sequence $(\alpha_0, \ldots, \alpha_n)$ of reals, the *number of sign variations* is the number of integers i such that there exists an integer j, $0 \le j < i$, with $\alpha_j\alpha_{j+k} \ge 0$, $k = 1, \ldots, i - j - 1$ and $\alpha_i\alpha_j < 0$.

Theorem 2 (Descartes) [23] *The number of sign variations in the coefficients of a univariate polynomial in* $\mathbb{R}[x]$ *exceeds the number of positive real roots by an even quantity, possibly 0.*

The following result follows easily, cf. e.g. [24, Ch.7, Exer.1.3]: For a polynomial in $\mathbb{R}[x]$ of degree d, the number of sign variations in its coefficient sequence gives precisely the number of positive roots assuming no root equals zero and there are d real roots.

Case	Number of positive roots of t	Root ordering
1	4	$x_2^- < x_2^+ < x_1^- < x_1^+$
2	3	$x_2^- < x_1^- < x_2^+ < x_1^+$
3a	2	$x_2^- < x_1^- < x_1^+ < x_2^+$
3b	2	$x_1^- < x_2^- < x_2^+ < x_1^+$
4	1	$x_1^- < x_2^- < x_1^+ < x_2^+$
5	0	$x_1^- < x_1^+ < x_2^- < x_2^+$

Table 1
The 6 different orderings for the roots of two quadratic polynomials, assuming that the roots are distinct; we denote by x_τ^+ the large root of f_τ, $\tau = 1, 2$.

Tables 1 and 2 show the possible root orderings and the number of positive roots for t when the coefficients of R take all possible signs. It is clear how

R_4	R_3	R_2	R_1	R_0	#roots > 0	case
$+$	J	$(+) + K$	JK	$(+) - JJ'$		
$+$	$-$	$-$	$-$	$-$	1	infeasible
$+$	$-$	$-$	$-$	$+$	2	infeasible
$+$	$-$	$-$	$+$	$-$	3	2 $(x_2^- < x_1^- < x_2^+ < x_1^+)$
$+$	$-$	$-$	$+$	$+$	2	3
$+$	$-$	$+$	$-$	$-$	3	2 $(x_2^- < x_1^- < x_2^+ < x_1^+)$
$+$	$-$	$+$	$-$	$+$	4	1 $(x_2^- < x_2^+ < x_1^- < x_1^+)$
$+$	$-$	$+$	$+$	$-$	3	2 $(x_2^- < x_1^- < x_2^+ < x_1^+)$
$+$	$-$	$+$	$+$	$+$	2	3
$+$	$+$	$-$	$-$	$-$	1	4 $(x_1^- < x_2^- < x_1^+ < x_2^+)$
$+$	$+$	$-$	$-$	$+$	2	3
$+$	$+$	$-$	$+$	$-$	3	infeasible
$+$	$+$	$-$	$+$	$+$	2	infeasible
$+$	$+$	$+$	$-$	$-$	1	4 $(x_1^- < x_2^- < x_1^+ < x_2^+)$
$+$	$+$	$+$	$-$	$+$	2	3
$+$	$+$	$+$	$+$	$-$	1	4 $(x_1^- < x_2^- < x_1^+ < x_2^+)$
$+$	$+$	$+$	$+$	$+$	0	5 $(x_1^- < x_1^+ < x_2^- < x_2^+)$

Table 2
Different cases according to the coefficient signs in the resultant, assuming $\alpha_\tau > 0$. Some of the combinations for the coefficient signs are not possible (e.g., if $R_3 < 0$ and $R_2 < 0$, then necessarily $K < 0$, which implies that $R_1 > 0$); these cases are denoted by the keyword "infeasible".

to deduce the sign of $t = x_1 - x_2$, when there are 0, 1, 3, or 4 positive roots. However, when there are two positive and two negative roots, two cases are possible. So an additional test is necessary, which can be reduced to the sign of $E := \Delta_1\alpha_2^2 - \Delta_2\alpha_1^2$.

4.2 Sturm sequences

This section surveys Sturm sequences; further details can be found in, e.g., [24]. This theory shall be applied in Section 8.5.3.

Given univariate polynomials $P_0, P_1 \in \mathbb{R}[x]$, their *Sturm sequence* is any (pseudo-remainder) sequence P of polynomials $P_0, P_1, \ldots, P_n \in \mathbb{R}[x],\ n \geq 1$ such that $\alpha P_{i-1} = T_i P_i + \beta P_{i+1},\ i = 1, \ldots, n-1$, for some $T_i \in \mathbb{R}[x], \alpha, \beta \in \mathbb{R}$, and $\alpha\beta < 0$. When a specific sequence of polynomials is understood and real number p is given, we shall denote by $V_P(p)$ the number of sign variations of the sequence of values obtained by evaluating the polynomials P_i at p.

Proposition 3 [24, Lect. VII, §3] *For relatively prime polynomials $A, B \in \mathbb{R}[x]$, where A is assumed square-free, consider any Sturm sequence P of*

$f_2(x_1^+)$	$f_2(x_1^-)$	$f_2'(x_1^+)$	$f_2'(x_1^-)$	sign(J)	Case
$-$	$-$	any	any	any	3a ($x_2^- < x_1^- < x_1^+ < x_2^+$)
$-$	$+$	any	$-$	$+$	4 ($x_1^- < x_2^- < x_1^+ < x_2^+$)
$-$	$+$	any	$+$		infeasible
$+$	$-$	$-$	any		infeasible
$+$	$-$	$+$	any	$-$	2 ($x_2^- < x_1^- < x_2^+ < x_1^+$)
$+$	$+$	$-$	$-$	$+$	5 ($x_1^- < x_1^+ < x_2^- < x_2^+$)
$+$	$+$	$-$	$+$		infeasible
$+$	$+$	$+$	$-$	any	3b ($x_1^- < x_2^- < x_2^+ < x_1^+$)
$+$	$+$	$+$	$+$	$-$	1 ($x_2^- < x_2^+ < x_1^- < x_1^+$)

Table 3
Cases according to the first 4 signs, for $\alpha_\tau, \Delta_\tau > 0$; sign$(J)$ is shown if it can be derived from the first 4 signs.

$A, A'B$. Then for any $p < q$ non-roots of A, it holds that

$$V_P(p) - V_P(q) = \sum_{A(\rho)=0,\ p<\rho<q} sign(B(\rho)).$$

The Sturm sequence here may be $(A, A'B, -A, \dots)$.

Ordering the roots of the f_τ can be reduced to deciding the sign of f_2 and f_2', say, evaluated at the root of f_1 which interests us (see Table 3). This is a direct application of Proposition 3, since we assume that the f_1 and f_2 have no common roots and that $\Delta_\tau > 0$, $\tau = 1, 2$. The case $\Delta_\tau = 0$ can be treated easily, since in this case we have explicit expressions for the double root of f_τ. The Sturm sequence $(P_i)_i$ of f_1 and $f_1' f_2$ is:

$$\begin{aligned}
P_0(x) &= f_1(x)\\
P_1(x) &= f_1'(x) f_2(x)\\
P_2(x) &= -f_1(x)\\
P_3(x) &= -2\alpha_1[(\alpha_1 K + 2\alpha_2\Delta_1)x + (\gamma_1 J - \alpha_1 J')]\\
P_4(x) &= -\alpha_1\Delta_1(\alpha_1 K + 2\alpha_2\Delta_1)^2(G^2 - 4JJ')
\end{aligned}$$

Similarly, the Sturm sequence $(Q_i)_i$ of f_1 and $f_1' f_2'$ is:

$$\begin{aligned}
Q_0(x) &= f_1(x)\\
Q_1(x) &= f_1'(x) f_2'(x)\\
Q_2(x) &= \alpha_2(-Jx + L)\\
Q_3(x) &= 4\alpha_2\Delta_1\alpha_1^2 J^2(\alpha_1\Delta_2 + \alpha_2 K)
\end{aligned}$$

where $L = \beta_1\beta_2 - \alpha_2\gamma_1$. Interestingly, the same quadratic invariants are encountered in the Sturm sequence approach; here we also have to deal with cubic invariants. Moreover, the constant term in the resultant is precisely the factor of highest degree to be tested in the Sturm sequence. If we are interested in the larger root of f_1, it is possible to apply Proposition 3 by choosing

$$p = \frac{\beta_1}{\alpha_1},\ q = \infty,$$

since they clearly avoid the roots of f_1 and their interval includes only the large root. For bounding the small root, use $-q, p$.

4.3 The inversion approach

In this section we show how to compute the tritangent CCW-Apollonius circle C_{ijk} corresponding to the triplet B_i, B_j, B_k which touches the sites B_ν, $\nu = i, j, k$, in the order $\{i, j, k\}$, when we walk on the boundary of C_{ijk} in the counter-clockwise sense. We call our approach the *inversion approach*. This is because we use an inversion mapping to transform the problem of computing a circle commonly tangent to three sites to that of computing a line co-tangent to two sites.

Let $\mathcal{Z}$ be the (complex) plane that contains the sites B_ν, $\nu = i, j, k$. Let B^*_ν, $\nu = i, j, k$ be the sites with centers the centers of the B_ν's and radii $r^*_\nu = r_\nu - r_i$. Clearly, the site B_i has now been reduced to the point b_i, whereas the remaining two sites may have negative radius. We call the plane that contains the sites B^*_ν, $\nu = i, j, k$ the $\mathcal{Z}^*$-plane. Consider the standard inversion mapping (cf. [25])

$$W(z) = \frac{z - z_i}{|z - z_i|^2} \tag{2}$$

between the complex plane $\mathcal{Z}^*$ and the complex plane $\mathcal{W}$, where z_i is the point b_i and $|z|$ stands for the norm of $z \in \mathbb{C}$. This mapping maps circles on the $\mathcal{Z}^*$-plane that do not pass through z_i to circles on the $\mathcal{W}$-plane, and circles that pass through z_i on the $\mathcal{Z}^*$-plane to lines on the $\mathcal{W}$-plane (cf. [25]).

Let C^*_{ijk} be the tritangent CCW-Apollonius circle of B^*_ν, $\nu = i, j, k$. To simplify our notation we drop the subscripts of C_{ijk} and C^*_{ijk}. Thus in the sequel C is in fact C_{ijk}, and C^* is in fact C^*_{ijk}. Let $P = (x, y)$ be the center of C, and let r be its radius. The sites B^*_ν, $\nu = j, k$, are transformed to the sites $W_\nu = \{(u_\nu, v_\nu), \rho_\nu\}$, $\nu = j, k$, on the $\mathcal{W}$-plane, where

$$u_\nu = \frac{x^*_\nu}{p^*_\nu}, \quad v_\nu = \frac{y^*_\nu}{p^*_\nu}, \quad \rho_\nu = \frac{r^*_\nu}{p^*_\nu}, \qquad \nu = j, k,$$

$$x^*_\nu = x_\nu - x_i, \quad y^*_\nu = y_\nu - y_i, \quad p^*_\nu = (x^*_\nu)^2 + (y^*_\nu)^2 - (r^*_\nu)^2, \qquad \nu = j, k. \tag{3}$$

Note that since the sites B^*_ν, $\nu = j, k$, are visible, we have $p^*_\nu > 0$, $\nu = j, k$. Moreover, the sign of the radii ρ_ν, $\nu = j, k$, is the same as the sign of the radii

r^*_ν, $\nu = j, k$.

The site C^* on the $\mathcal{Z}^*$-plane is transformed to a line L on the $\mathcal{W}$-plane. Let $\bar{a}_{ijk}u + \bar{b}_{ijk}v + \bar{c}_{ijk} = 0$, $\bar{a}^2_{ijk} + \bar{b}^2_{ijk} = 1$, be the equation of the line L. Since the sites B^*_ν, $\nu = j, k$, are tangent to C^* on the $\mathcal{Z}^*$-plane, the sites W_ν, $\nu = j, k$, are tangent to the line L on the $\mathcal{W}$-plane (the inversion transformation preserves tangency and containment relations). We also required that the tritangent circle C has the correct orientation, i.e., as we walk on the boundary of C in counter-clockwise order, we have the sites B_i, B_j and B_k to the right and we touch the sites B_ν, $\nu = i, j, k$, in this order. The ordering does not change when we reduce the problem of finding C to that of finding C^*. However, the sites B^*_ν may now be on different sides of C^*, depending on the sign of r^*_ν, $\nu = j, k$. In particular, if r^*_ν is positive the site B^*_ν is to the right of C^*, whereas if $r^*_\nu < 0$, the site B^*_ν is to the left of C^*. If r^*_ν is zero, then we can choose any of the two sides without loss of generality.

In the $\mathcal{W}$-plane this requirement can be stated as follows: as we walk on the line L, the sites W_j, W_k must be on the same side of L as in the $\mathcal{Z}^*$-plane. This can be achieved by requiring that the vector $(\bar{a}_{ijk}, \bar{b}_{ijk})$ is oriented in such a way that the positive half-plane with respect to L contains the circle(s) with ρ_ν positive and the negative half-plane contains the circle(s) with ρ_ν negative. The vector that is parallel to L in the direction that we traverse L is the vector $(\bar{b}_{ijk}, -\bar{a}_{ijk})$. On the $\mathcal{Z}^*$-plane we also required that we touch the sites B_ν, $\nu = i, j, k$, in this order. This requirement in the $\mathcal{W}$-plane means that as we walk on L in the direction $(\bar{b}_{ijk}, -\bar{a}_{ijk})$, we first touch W_j and then W_k. This is equivalent to requiring that the projection of the vector $(u_k - u_j, v_k - v_j)$ on the line L is positive. This is exactly the problem of Section 7.2, where now we also allow sites of negative radius. Using Section 7.2, we get:

$$\bar{a}_{ijk} = \frac{D^u_{jk}D^\rho_{jk} + D^v_{jk}\sqrt{\Delta_{jk}}}{{D^u_{jk}}^2 + (D^v_{jk})^2}, \qquad \bar{b}_{ijk} = \frac{D^v_{jk}D^\rho_{jk} - D^u_{jk}\sqrt{\Delta_{jk}}}{(D^u_{jk})^2 + (D^v_{jk})^2}, \tag{4}$$

$$\bar{c}_{ijk} = \frac{D^u_{jk}D^{u\rho}_{jk} + D^v_{jk}D^{v\rho}_{jk} + D^{uv}_{jk}\sqrt{\Delta_{jk}}}{(D^u_{jk})^2 + (D^v_{jk})^2}, \qquad \Delta_{jk} = (D^u_{jk})^2 + (D^v_{jk})^2 - (D^\rho_{jk})^2, \tag{5}$$

where

$$D^s_{\lambda\nu} = \begin{vmatrix} s_\lambda & 1 \\ s_\nu & 1 \end{vmatrix}, \qquad D^{st}_{\lambda\nu} = \begin{vmatrix} s_\lambda & t_\lambda \\ s_\nu & t_\nu \end{vmatrix}, \qquad s, t \in \{u, v, \rho\}, \quad \lambda, \nu \in \{j, k\}.$$

By using the inverse of the transformation (2), we can easily verify that the line $\bar{a}_{ijk}u + \bar{b}_{ijk}v + \bar{c}_{ijk} = 0$ is transformed to the circle:

$$(x + \frac{\bar{a}_{ijk}}{2\bar{c}_{ijk}} - x_i)^2 + (y + \frac{\bar{b}_{ijk}}{2\bar{c}_{ijk}} - y_i)^2 = \frac{1}{4\bar{c}_{ijk}^2}, \tag{6}$$

provided of course that $\bar{c}_{ijk} \neq 0$. If $\bar{c}_{ijk} = 0$, the line $\bar{a}_{ijk}u + \bar{b}_{ijk}v + \bar{c}_{ijk} = 0$ is mapped to the line:

$$\bar{a}_{ijk}(x - x_i) + \bar{b}_{ijk}(y - y_i) = 0,$$

which geometrically means that the three sites B_ν, $\nu = i, j, k$, are in a degenerate condition and instead of having two tritangent Apollonius circles, they have a common tritangent line. In other words in this case the center of the tritangent Apollonius circle is at infinity. Considering again the non-degenerate case, equation (6) is the equation of the circle C^*. Hence the center of C is

$$(x, y) = (-\frac{\bar{a}_{ijk}}{2\bar{c}_{ijk}} + x_i, -\frac{\bar{b}_{ijk}}{2\bar{c}_{ijk}} + y_i) \tag{7}$$

whereas the radius r of C is the radius of C^* reduced by r_i, i.e.,

$$r = \frac{1}{2\bar{c}_{ijk}} - r_i.$$

Here we assumed that c is positive. This is a valid hypothesis because, as we will see in Section 8.2, it is equivalent to requiring that the tritangent Apollonius circle exists.

The expressions of $\bar{a}_{ijk}$, $\bar{b}_{ijk}$ and $\bar{c}_{ijk}$ in terms of the original coordinates are:

$$\bar{a}_{ijk} = \frac{E_{ijk}^{xp}E_{ijk}^{rp} + E_{ijk}^{yp}\sqrt{\Gamma_{ijk}}}{(E_{ijk}^{xp})^2 + (E_{ijk}^{yp})^2}, \qquad \bar{b}_{ijk} = \frac{E_{ijk}^{yp}E_{ijk}^{rp} - E_{ijk}^{xp}\sqrt{\Gamma_{ijk}}}{(E_{ijk}^{xp})^2 + (E_{ijk}^{yp})^2}, \tag{8}$$

$$\bar{c}_{ijk} = \frac{E_{ijk}^{xp}E_{ijk}^{xr} + E_{ijk}^{yp}E_{ijk}^{yr} + E_{ijk}^{xy}\sqrt{\Gamma_{ijk}}}{(E_{ijk}^{xp})^2 + (E_{ijk}^{yp})^2}, \tag{9}$$

where

$$E_{\lambda\mu\nu}^{st} = \begin{vmatrix} s_\mu^* & t_\mu^* \\ s_\nu^* & t_\nu^* \end{vmatrix}, \qquad s, t \in \{x, y, r, p\}, \quad \lambda, \mu, \nu \in \{i, j, k\},$$

and

$$\Gamma_{ijk} = (E_{ijk}^{xp})^2 + (E_{ijk}^{yp})^2 - (E_{ijk}^{rp})^2. \tag{10}$$

5 Dealing with degeneracies

We handle inputs with degeneracies by the standard conceptual infinitesimal perturbation method. This involves no actual computation with the positive infinitesimal variable ϵ; its use merely specifies the actual quantities that must be computed and tested. Moreover, our perturbations do not increase the asymptotic complexity of the algorithm. For general information on symbolic perturbations one may consult, e.g., [26,27].

In the dynamic algorithm described above the Apollonius diagram is represented by its dual graph. *Degenerate* instances are precisely those that lead to non-triangular faces, or to tritangents on the convex hull of $\mathcal{B}$. The possible degenerate configurations, from the point of view of analyzing the predicates, are quite numerous. Elaborating on all of them is possible but of no interest. All degenerate cases arise from two basic configurations: either four sites with a common tangent Apollonius circle, or three sites that are tangent to the same line and lie on the same halfspace with respect to that line. As we will see below, the first degeneracy is handled really easily: we basically consider such a configuration as being the same with the case were the fourth site does not intersect the Apollonius circle of the first three. The second degeneracy is the one that calls for our symbolic perturbation, and may be analyzed as follows.

It is required that all sites on the convex hull are connected to B_∞ in the dual graph; this is a *canonical* configuration, since near the convex hull it does not depend on the order of insertion. The canonicity requirement is achieved by means of a *local* infinitesimal perturbation scheme, which resolves the degenerate cases near the convex hull. Locality has a twofold sense. First, the perturbation applies only when a specific subpredicate is considered, namely DistanceFromBitangent (see Section 3.3), and does not constitute a preprocessing step that modifies the entire set of input sites. In addition, a site is perturbed in a certain way during an evaluation of this subpredicate, while it might be perturbed in a different way during another evaluation. Our scheme guarantees, however, that the resulting triangulated dual graph is coherent (or consistent) and correct. More precisely, when DistanceFromBitangent is called, we shall consider perturbed sites $B^\epsilon = \{b, r - \epsilon\,\tau\}$, where τ is some real quantity determined by the problem at hand and $\epsilon \rightarrow 0^+$ is the infinitesimal indeterminate. Note that the implementation does not have to introduce ϵ; this is simply used for the purposes of the analysis and in order to derive the quantities to be tested.

Our perturbation does not affect the dual graph of the internal sites. In case of degeneracies, the non-triangular faces of the dual graph can be triangulated in an arbitrary way, because there is no canonicity requirement in the interior of the hull. The dual graph itself depends on the order of insertion away from the convex hull. One may consider that an implicit perturbation is applied, since it has been incorporated in the definitions of shadow and conflict region (e.g., the cases $\delta(C_{ijk}, B) > 0$ and $\delta(C_{ijk}, B) = 0$ are identical).

When deletions are allowed, this calls for some care at the removal stage. Let B be a visible site we want to delete from the Apollonius diagram and let us assume temporarily that none of the sites in $L_h(B)$ will become visible after the deletion of B. The deletion procedure described above essentially simulates the insertion of B in the Apollonius diagram of its neighbors. In fact by simulating this insertion it is implicitly assumed that B is the last site that would have been inserted in $\mathcal{V}(\mathcal{B}_\gamma \cup \{B\})$. If the Apollonius cell of B has only Apollonius vertices of degree 3, then the order in which B is inserted in $\mathcal{V}(\mathcal{B}_\gamma \cup \{B\})$ is not important; the star of B in the Apollonius graph $\mathcal{D}(\mathcal{B}_\gamma \cup \{B\})$ is uniquely defined. On the contrary, when the Apollonius cell of B has vertices of degree higher than 3, then the star of B in $\mathcal{D}(\mathcal{B}_\gamma \cup \{B\})$ depends on the fact that B is the last site inserted. In fact, the way we construct the Apollonius graph, the degree of B in $\mathcal{D}(\mathcal{B}_\gamma \cup \{B\})$ is the minimum one among all possible valid triangulations of the dual of the Apollonius diagram for the set $\mathcal{B}_\gamma \cup \{B\}$. Since in the original Apollonius graph $\mathcal{D}(\mathcal{B})$, B may not be the last site inserted, the star of B in $\mathcal{D}(\mathcal{B})$ and $\mathcal{D}(\mathcal{B}_\gamma \cup \{B\})$ may differ.

One way to remedy this is to perform the deletion of B in two stages: at first we modify the Apollonius graph $\mathcal{D}(\mathcal{B})$ so as to simulate that B is the last site inserted. In fact this amounts to minimizing B's degree in the Apollonius graph; minimization is understood over all possible valid Apollonius graphs. Note that it is possible to impose the desired dual graph, since ties can be broken at will in the interior of the hull. The second stage amounts to performing the deletion as it has already been described. Minimizing the degree of B essentially means flipping some edges in the Apollonius graph $\mathcal{D}(\mathcal{B})$ before proceeding into the second stage of the deletion. The edges in the Apollonius graph that are flipped are edges added to the actual dual of the Apollonius diagram to make the Apollonius graph consisting of only triangular faces. The edges in the Apollonius diagram that they correspond to can be viewed as degenerate edges, i.e., edges whose endpoints coincide. Flipping an edge in this context corresponds to a different possible way of triangulating the non-triangular faces of the dual of the Apollonius diagram. From the point of view of the predicates this calls for one more predicate, namely, determine if an edge of the Apollonius diagram, defined by four sites, is degenerate.

Let us analyze a concrete example that illustrates the problem and how we resolve it. Consider the set $\mathcal{B}$ of 9 point sites B_i, $i = 1, \ldots, 9$, shown in Fig. 5 (we consider point sites for simplicity; the situation for non-point sites is entirely analogous). The first 5 sites are cocircular, and if we insert them in order of increasing index, the resulting Apollonius graph will be the one

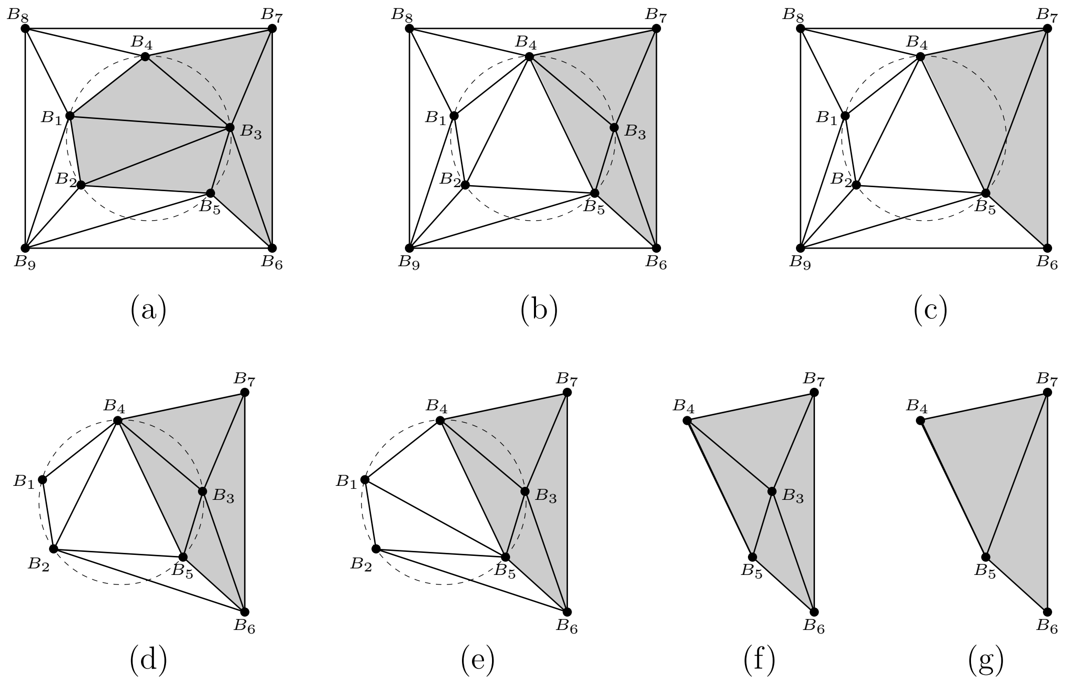


Fig. 5. The deletion procedure in the presence of degeneracies away from the convex hull. (a): a set of 9 point sites $\mathcal{B} = \{B_i, i = 1, \ldots, 9\}$, with 5 cocircular sites $(B_i, i = 1, \ldots, 5)$, and the corresponding Apollonius graph when sites are inserted in order of increasing index. (b): the Apollonius graph after minimizing the degree of B_3. (c): the Apollonius graph after the deletion of B_3. (d) and (e): the two possible Apollonius graphs for the set $\mathcal{B}_\gamma \cup \{B_3\}$ from Subfigure (a). (f): the Apollonius graph for the set $\mathcal{B}_\gamma \cup \{B_3\}$ from Subfigure (b). (g): the Apollonius graph of the set $\mathcal{B}_\gamma$ from Subfigure (b). For simplicity, the edges to the site at infinity are not shown. The star of B_3 is shown in gray.

shown in Fig. 5(a). Suppose now that we want to delete B_3. Our deletion procedure would require that we construct the Apollonius graph of the set $\mathcal{B}_\gamma$ of neighbors of B_3; here $\mathcal{B}_\gamma = \{B_1, B_2, B_4, B_5, B_6, B_7\}$. Then we would have to superimpose $\mathcal{D}(\mathcal{B})$ with $\mathcal{D}(\mathcal{B}_\gamma)$ and retriangulate the star of B_3 in $\mathcal{D}(\mathcal{B})$, using the Apollonius graph $\mathcal{D}(\mathcal{B}_\gamma)$. This procedure is based on the assumption that the boundaries of the stars of B_3 in $\mathcal{D}(\mathcal{B})$ and $\mathcal{D}(\mathcal{B}_\gamma \cup \{B\})$ are identical. In our example this is not the case. The two possible Apollonius graphs of the set $\mathcal{B}_\gamma \cup \{B\}$ are shown in Figs. 5(d) and 5(e): the star of B_3 in either of these graphs differs from its star in $\mathcal{D}(\mathcal{B})$ (recall that we require B_3 to be the last one inserted in $\mathcal{D}(\mathcal{B}_\gamma \cup \{B\})$). Notice also that the degree of B_3 in Figs. 5(d) and 5(e) has the property that it is minimal among all possible valid Apollonius graphs of the set $\mathcal{B}_\gamma \cup \{B\}$ (or $\mathcal{B}$). The afore-mentioned observations motivated our two-stage approach. We first minimize the degree of B_3 in $\mathcal{D}(\mathcal{B})$: the resulting Apollonius graph is shown in Fig. 5(b) (assuming that the edge b_1b_3 is flipped before the edge b_2b_3). The new set of neighbors of B in $\mathcal{D}(\mathcal{B})$ is $\mathcal{B}_\gamma = \{B_4, B_5, B_6, B_7\}$. We then perform the actual deletion procedure. The star of B_3 is now identical in $\mathcal{D}(\mathcal{B})$ and $\mathcal{D}(\mathcal{B}_\gamma \cup \{B\})$ (cf. Figs. 5(b) and 5(f)) and we can retriangulate the star of B_3 by superimposing $\mathcal{D}(\mathcal{B})$ and $\mathcal{D}(\mathcal{B}_\gamma)$ (cf. Figs. 5(c) and 5(g)).

6 The first three predicates

6.1 The SideOfBisector *predicate.*

Let $B_\nu = \{(x_\nu, y_\nu), r_\nu\}$, $\nu = i, j$, be two sites and let B be a query site. The SideOfBisector(b, B_i, B_j) predicate is equivalent to finding the sign of the quantity Q_{SB}, where

$$\begin{aligned} Q_{SB} &:= \delta(b, B_i) - \delta(b, B_j) \\ &= \sqrt{(x-x_i)^2 + (y-y_i)^2} - \sqrt{(x-x_j)^2 + (y-y_j)^2} - r_i + r_j. \end{aligned} \tag{11}$$

Q_{SB} is a quantity of the form $A_0 + A_1\sqrt{B_1} + A_2\sqrt{B_2}$, which we can rewrite as $A_0' + A_1'\sqrt{B'}$, where $A_0' = A_0 + A_1\sqrt{B_1}$, $A_1' = A_2$ and $B' = B_2$. The sign of Q_{SB} can be determined easily if we know how to determine the sign of quantities of the form $X_0 + X_1\sqrt{Y}$. The latter can be done easily using the following formula:

$$\operatorname{sign}(X_0 + X_1\sqrt{Y}) = \operatorname{sign}(\operatorname{sign}(X_0)X_0^2 + \operatorname{sign}(X_1)X_1^2 Y) \tag{12}$$

In particular, we need first check if Y is zero; if this is the case, then $\operatorname{sign}(X_0 + X_1\sqrt{Y}) = \operatorname{sign}(X_0)$. If $Y > 0$ we need to check if $\operatorname{sign}(X_0)$ and $\operatorname{sign}(X_2)$ are the same. Clearly, in this case, $\operatorname{sign}(X_0 + X_1\sqrt{Y}) = \operatorname{sign}(X_0) = \operatorname{sign}(X_1)$. If $\operatorname{sign}(X_0) \neq \operatorname{sign}(X_1)$ we need to distinguish between two cases for X_0. If $X_0 = 0$, then $\operatorname{sign}(X_0 + X_1\sqrt{Y}) = \operatorname{sign}(X_1)$. Otherwise, $\operatorname{sign}(X_0 + X_1\sqrt{Y}) = \operatorname{sign}(X_0)\operatorname{sign}(X_0^2 - X_1^2 Y)$. We can summarize the procedure, described above in words, as follows:

$$\operatorname{sign}(X_0 + X_1\sqrt{Y}) = \begin{cases} \operatorname{sign}(X_0) & \text{if } Y = 0 \\ \operatorname{sign}(X_0) & \text{if } \operatorname{sign}(X_0) = \operatorname{sign}(X_1) \\ \operatorname{sign}(X_1) & \text{if } X_0 = 0 \\ \operatorname{sign}(X_0)\operatorname{sign}(X_0^2 - X_1^2 Y) & \text{otherwise} \end{cases}. \tag{13}$$

In order to compute the sign of Q_{SB}, we need to apply (13) recursively. Conceptually, we firstly apply it with $X_0 = A_0 + A_1\sqrt{B_1}$, $X_1 = A_2$ and $Y = B_2$. We recursively apply it to compute $\operatorname{sign}(X_0)$. Finally, we may need to apply it once more to compute the sign of $X_0^2 - X_1^2 Y = A_0^2 + A_1^2 B_1 - A_2^2 B_2 + 2A_0A_1\sqrt{B_1}$. This amounts to computing the sign of the quantity $(A_0^2 + A_1^2 B_1 - A_2^2 B_2)^2 - 4A_0^2A_1^2B_1$, which is homogeneous with respect to its algebraic degree in the input quantities, and in particular its algebraic degree is 4 (the algebraic degrees of A_0, A_1, A_2, B_1 and B_2 are 1, 0, 0, 2 and 2, respectively.) We thus deduce that the algebraic degree of the SideOfBisector predicate is 4 (in the input quantities).

6.2 *The* IsHidden *predicate.*

Let B_i be a site and B be a query site. We want to determine if B is contained inside B_i. This is equivalent to determining the sign of the quantity Q_H, where:

$$Q_H := \delta(B, B_i) + 2r = \sqrt{(x - x_i)^2 + (y - y_i)^2} + r - r_i.$$

We can determine the sign of Q_H by using relation (12). Thus, the algebraic degree of the IsHidden(B, B_i) predicate is 2.

6.3 *The* IsIntersecting *predicate.*

Let B_i and B_j be two sites. IsIntersecting(B_i, B_j) is equivalent to determining the sign of the quantity Q_X, where:

$$Q_X := \delta(B_i, B_j) = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2} - r_i - r_j.$$

We can again determine sign(Q_X) by using relation (12). The algebraic degree of IsIntersecting is clearly 2.

Collecting the results above we get:

Lemma 4 *The algebraic degrees of the* SideOfBisector*,* IsHidden *and* IsIntersecting *predicates is 4, 2 and 2, respectively.*

In the remaining sections we describe how to evaluate the remaining three predicates. The sites involved in the computations that follow are considered visible. Finally, we do not consider the EdgeConflictType predicate directly, but rather its two versions, namely InfiniteEdgeConflictType and FiniteEdgeConflictType, depending on whether the Apollonius edge tested lies on an infinite or finite bisector, respectively.

7 The InfiniteEdgeConflictType predicate

Determining the type of the conflict region of an infinite Apollonius edge $\alpha_{i\infty}^{k\ell}$ with respect to a site B_m reduces to determining the type of intersection of the circular arcs $\alpha_{i\infty}^{k\ell}$ and $S_{i\infty}(B_m)$ on ∂B_i. Let $p_{\ell i}$, p_{ik} denote the points of tangency of $C_{i\ell\infty}$, $C_{ki\infty}$ with B_i respectively. Let also q_1 and q_2 denote the endpoints of the circular arc $S_{i\infty}(B_m)$. We can decide InfiniteEdgeConflictType as follows (cf. Fig. 6).

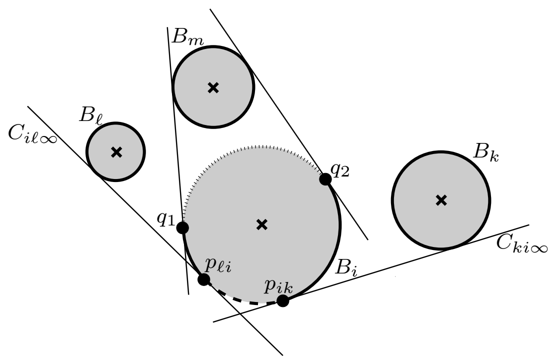
We first check if $p_{\ell i}$ and p_{ik} lie in $S_{i\infty}(B_m)$ using the DistanceFromBitangent subpredicate, which checks the sign of the distance of the site B_λ from the line $L_{\mu\nu}$ bitangent to the sites B_μ and B_ν ($L_{\mu\nu} \equiv C_{\nu\mu\infty}$). In terms of the conflict region of B_m w.r.t. $\alpha_{i\infty}^{k\ell}$, DistanceFromBitangent tells us if an endpoint of $\alpha_{i\infty}^{k\ell}$ is in the shadow region $S_{i\infty}(B_m)$. Its degree is 6 (cf. Section 7.2). If exactly one of $p_{\ell i}$ and p_{ik} belongs to $S_{i\infty}(B_m)$ we are done. Otherwise, we need to determine if the arc $\widehat{p_{\ell i}p_{ik}}$ is entirely inside $S_{i\infty}(B_m)$. This is done by checking if the point q_1 (or q_2) lies inside $\widehat{p_{\ell i}p_{ik}}$, which calls for the InsideCircularArc subpredicate. We discuss this subpredicate in Section 7.1 and show that its degree is also 6.

In the above analysis we implicitly assumed that our data are in non-degenerate position, i.e., the signed distance of B_m from the bitangent lines $L_{\ell i}$ and L_{ik} is non-zero. Suppose that $\delta(B, L_{\ell i}) = 0$. Then whether or not $p_{\ell i}$ or p_{ik} is in conflict with B_m depends on whether B_m is actually on the convex hull of the set of sites in between B_ℓ and B_i, or B_i and B_k, respectively. Let t_ℓ, t_i and t_m be the points of tangency of B_ℓ, B_i and B_m with $L_{\ell i}$ (clearly $t_i \equiv p_{\ell i}$). Then B_m is in conflict with $p_{\ell i}$ if and only if t_m lies in the interior of the segment $t_\ell t_i$ (since we have visible sites the points t_ℓ, t_i, t_m are distinct). This is discussed in Section 7.2. We shall compute the distance of a perturbed site $B_m^\epsilon = \{b_m, r_m - \epsilon\,(\overrightarrow{t_\ell t_m} \cdot \overrightarrow{t_i t_m})\}$ with respect to $L_{\ell i}$ (cf. Fig. 7). The degree of DistanceFromBitangent does not change due to this perturbation.

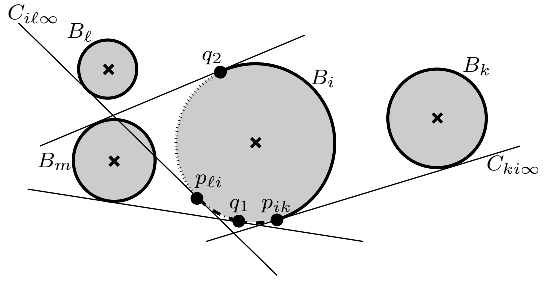
Lemma 5 *The algebraic degree of the* InfiniteEdgeConflictType *predicate is 6.*

7.1 The InsideCircularArc *subpredicate*

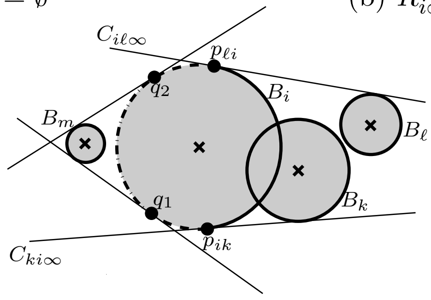
Consider a circle C. Given a point p on C, we denote as $\vec{p}$ the unit vector in the direction of $\overrightarrow{cp}$, where c is the center of C. Given two points p and q on C we denote as $\widehat{pq}$ the counter-clockwise arc on C that starts from p and ends at q. Consider now a third point r on C. We want to determine if r is inside the arc $\widehat{pq}$. This is the InsideCircularArc subpredicate. Suppose that we have the primitive $\chi_2(\vec{p}, \vec{q})$ which returns the sign of the z-coordinate of the cross



(a) $\tilde{R}^{k\ell}_{i\infty}(B) = \emptyset$



(b) $\tilde{R}^{k\ell}_{i\infty}(B) = (\eta', c_{i\ell\infty}]$



(c) $\tilde{R}^{k\ell}_{i\infty}(B) = (\xi', \eta')$

Fig. 6. Three of the 6 possible (non-degenerate) cases for the conflict region $R^{k\ell}_{i\infty}(B)$ of an Apollonius edge $\alpha^{k\ell}_{i\infty}$ lying on an infinite bisector. The map of the edge $\alpha^{k\ell}_{i\infty}$ on ∂B_i is depicted as a dashed arc. The shadow region $S_{i\infty}(B)$ is depicted as a dotted arc. The dash-dotted arc in Subfigures (b) and (c) is the intersection $S_{i\infty}(B) \cap \alpha^{k\ell}_{i\infty}$.

product $\vec{p} \times \vec{q}$. Then we can easily decide the InsideCircularArc predicate as follows.

The first thing to do is to check if p and q are the same or antipodal points on C. This can be determined by looking at the signs of $\chi_2(\vec{p}, \vec{q})$ and $\chi_2(\vec{p}, \vec{q}^{\perp})$, where $\vec{q}^{\perp}$ is the vector $\vec{q}$ rotated counterclockwise by $\frac{\pi}{2}$. If $\chi_2(\vec{p}, \vec{q}) = 0$ and $\chi_2(\vec{p}, \vec{q}^{\perp}) > 0$, then the arc $\widehat{pq}$ has empty interior, whereas if $\chi_2(\vec{p}, \vec{q}) = 0$ and $\chi_2(\vec{p}, \vec{q}^{\perp}) < 0$, the arc $\widehat{pq}$ is a half-circle. In the first case we can further decide if r coincides with p and q or if it is outside $\widehat{pq}$ by looking at the signs $\chi_2(\vec{p}, \vec{r})$ and $\chi_2(\vec{p}, \vec{r}^{\perp})$. In the second case it suffices to look at the sign $\chi_2(\vec{p}, \vec{r})$. If the arc $\widehat{pq}$ has neither empty interior nor is a half-circle, we can answer the InsideCircularArc subpredicate by looking at the signs $\chi_2(\vec{p}, \vec{r})$ and $\chi_2(\vec{q}, \vec{r})$, and taking into account the sign $\chi_2(\vec{p}, \vec{q})$. Note that when $\chi_2(\vec{p}, \vec{q}) > 0$ (resp. $\chi_2(\vec{p}, \vec{q}) < 0$) the arc $\widehat{pq}$ is smaller (resp. greater) than $\frac{\pi}{2}$.

In our problem, the points p, q and r are tangent points on bitangent lines of two circles. This specific version of the χ_2 primitive has been studied and analyzed in [19]. A method to compute this primitive using quantities up to degree 6 is provided; as in our case, circles are assumed to be given by the coordinates of their center and their radius. Hence:

Lemma 6 *The* InsideCircularArc *subpredicate can be evaluated using quantities of algebraic degree up to 6.*

7.2 The DistanceFromBitangent *subpredicate*

Let $L := ax + by + c = 0$ be the equation of a line. The (signed) distance $\delta(B, L)$ of a site $B = \{(b_x, b_y), r\}$ from L is defined to be:

$$\delta(B, L) = \delta(b, L) - r, \qquad \delta(b, L) = \frac{ab_x + bb_y + c}{\sqrt{a^2 + b^2}}.$$

where $\delta(b, L)$ is the (signed) distance of b from the line L. Let B_i, B_j and B_k be three sites. Let L_{ij} denote the oriented bitangent line of B_i and B_j that has the following two properties:

(1) B_i and B_j are to the left of L_{ij}
(2) as we walk on L_{ij} in the *positive* direction, L_{ij} touches the two sites B_i and B_j in the order $\{i, j\}$

In this section we show how to compute the sign of the distance of $\delta(B_k, L_{ij})$ of B_k from L_{ij}.

Let $a_{ij}x + b_{ij}y + c_{ij} = 0$, $a_{ij}^2 + b_{ij}^2 = 1$ be the equation of the bitangent line L_{ij} that we seek. Since the sites B_i and B_j are tangent to L_{ij} and lie to the left of L_{ij}, we have that the (signed) distance of b_i and b_j from L_{ij} is equal to r_i and r_j, respectively. Hence

$$a_{ij}x_\lambda + b_{ij}y_\lambda + c_{ij} = r_\lambda, \qquad \lambda = i, j. \tag{14}$$

Since the sites are visible we must have that $(x_i - x_j)^2 + (y_i - y_j)^2 \neq 0$. We can then solve the system of linear equations (14), along with the quadratic equation $a_{ij}^2 + b_{ij}^2 = 1$, to get the following two solutions:

$$a_{ij} = \frac{D_{ij}^x D_{ij}^r \mp D_{ij}^y \sqrt{\Delta_{ij}}}{(D_{ij}^x)^2 + (D_{ij}^y)^2}, \qquad b_{ij} = \frac{D_{ij}^y D_{ij}^r \pm D_{ij}^x \sqrt{\Delta_{ij}}}{(D_{ij}^x)^2 + (D_{ij}^y)^2},$$
$$c_{ij} = \frac{D_{ij}^x D_{ij}^{xr} + D_{ij}^y D_{ij}^{yr} \mp D_{ij}^{xy} \sqrt{\Delta_{ij}}}{(D_{ij}^x)^2 + (D_{ij}^y)^2}, \tag{15}$$

where

$$D_{\lambda\nu}^s = \begin{vmatrix} s_\lambda & 1 \\ s_\nu & 1 \end{vmatrix}, \qquad D_{\lambda\nu}^{st} = \begin{vmatrix} s_\lambda & t_\lambda \\ s_\nu & t_\nu \end{vmatrix}, \qquad s, t \in \{x, y, r\}, \quad \lambda, \nu \in \{i, j\},$$

and

$$\Delta_{ij} = (D^x_{ij})^2 + (D^y_{ij})^2 - (D^r_{ij})^2.$$

Since we assumed that the sites B_i, B_j are mutually visible, we immediately get that $\Delta_{ij} > 0$. Therefore, we always have two real solutions for our system. This is nothing but an algebraic justification that two circles, such that none of the two is inside the other, have always two exterior bitangent lines.

The vector that is parallel to L_{ij} in the direction that we traverse L_{ij} is the vector $(b_{ij}, -a_{ij})$. The requirement that, as we walk on L_{ij} in the direction of $(b_{ij}, -a_{ij})$, we meet B_i and B_j in the order $\{i, j\}$ is equivalent to requiring that the projection of the vector $(x_j - x_i, y_j - y_i)$ on L_{ij} is positive. Algebraically this can be written as:

$$(b_{ij}, -a_{ij}) \cdot (x_j - x_i, y_j - y_i) > 0,$$

or equivalently,

$$(-b_{ij}, a_{ij}) \cdot (D^x_{ij}, D^y_{ij}) > 0.$$

Substituting the expressions for a_{ij} and b_{ij} in the above inequality we get:

$$-\frac{D^y_{ij}D^r_{ij} \pm D^x_{ij}\sqrt{\Delta_{ij}}}{(D^x_{ij})^2 + (D^y_{ij})^2}\, D^x_{ij} + \frac{D^x_{ij}D^r_{ij} \mp D^y_{ij}\sqrt{\Delta_{ij}}}{(D^x_{ij})^2 + (D^y_{ij})^2}\, D^y_{ij} > 0,$$

which reduces to the inequality:

$$\mp\sqrt{\Delta_{ij}} > 0.$$

Since $\Delta_{ij} > 0$, we deduce that the solution of interest is the solution:

$$a_{ij} = \frac{D^x_{ij}D^r_{ij} + D^y_{ij}\sqrt{\Delta_{ij}}}{(D^x_{ij})^2 + (D^y_{ij})^2}, \quad b_{ij} = \frac{D^y_{ij}D^r_{ij} - D^x_{ij}\sqrt{\Delta_{ij}}}{(D^x_{ij})^2 + (D^y_{ij})^2},$$
$$c_{ij} = \frac{D^x_{ij}D^{xr}_{ij} + D^y_{ij}D^{yr}_{ij} - D^{xy}_{ij}\sqrt{\Delta_{ij}}}{(D^x_{ij})^2 + (D^y_{ij})^2}. \tag{16}$$

Substituting a_{ij}, b_{ij} and c_{ij} from (16) in the expression for $\delta(B_k, L_{ij})$, and using the fact that $a^2_{ij} + b^2_{ij} = 1$, we get:

$$\delta(B_k, L_{ij}) = \frac{D^x_{ij}D^{xr}_{ijk} + D^y_{ij}D^{yr}_{ijk} + D^{xy}_{ijk}\sqrt{\Delta_{ij}}}{(D^x_{ij})^2 + (D^y_{ij})^2}, \tag{17}$$

where

$$D^{st}_{\lambda\mu\nu} = \begin{vmatrix} s_\lambda & t_\lambda & 1 \\ s_\mu & t_\mu & 1 \\ s_\nu & t_\nu & 1 \end{vmatrix}, \qquad s, t \in \{x, y, r\}, \quad \lambda, \mu, \nu \in \{i, j, k\}.$$

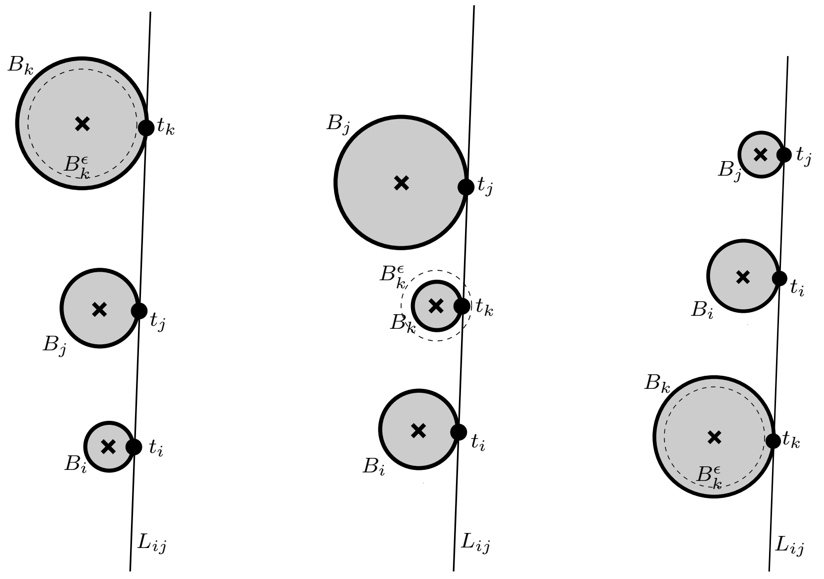


Fig. 7. The three degenerate cases for the DistanceFromBitangent subpredicate. Middle: the case $\overrightarrow{t_it_k} \cdot \overrightarrow{t_jt_k} < 0$ ($\delta(B^\epsilon_k, L_{ij}) < 0$). Left, Right: the two cases $\overrightarrow{t_it_k} \cdot \overrightarrow{t_jt_k} > 0$ ($\delta(B^\epsilon_k, L_{ij}) > 0$). The dashed circle denotes the perturbed site B^ϵ_k.

Clearly, the sign of $\delta(B_k, L_{ij})$ is the sign of $D^x_{ij}D^{xr}_{ijk} + D^y_{ij}D^{yr}_{ijk} + D^{xy}_{ijk}\sqrt{\Delta_{ij}}$, which can be computed using relation (12); its algebraic degree is 6, since the algebraic degrees of D^s_{ij}, D^{sr}_{ijk}, $s \in \{x, y\}$, D^{xy}_{ijk} and Δ_{ij} are 1, 2, 2 and 2, respectively. Hence,

Lemma 7 *The algebraic degree of the* DistanceFromBitangent *subpredicate is 6.*

We now apply our local perturbation scheme (cf. Fig. 7). We saw that if $\delta(B_k, L_{ij}) = 0$, then we need to compute the sign of the quantity $\overrightarrow{t_it_k} \cdot \overrightarrow{t_jt_k}$, where t_i, t_j, t_k, are the points of contact of B_ν, $\nu = i, j, k$, with the line L_{ij}. This is done as follows. Let $L^\perp_{\mu\nu}(p)$ be the line perpendicular to $L_{\mu\nu}$ through p. We check if t_k lies in the interior of t_it_j by computing the orientation of b_k with respect to the lines $L^\perp_{ij}(b_i)$ and $L^\perp_{ij}(b_j)$. These tests amount to computing the signs of $o_\nu = b_{ij}(x_\nu - x_k) - a_{ij}(y_\nu - y_k)$, $\nu = i, j$, which can be shown to be of degree 6. If o_ν are of the same (resp. opposite) sign then $\overrightarrow{t_it_k} \cdot \overrightarrow{t_jt_k} > 0$ (resp. $\overrightarrow{t_it_k} \cdot \overrightarrow{t_jt_k} < 0$) and $\delta(B^\epsilon_k, L_{ij}) > 0$ (resp. $\delta(B^\epsilon_k, L_{ij}) < 0$).

8 The FiniteEdgeConflictType predicate

Let $\alpha^{k\ell}_{ij}$ be an Apollonius edge and let B_m be our query site. The result of FiniteEdgeConflictType is the type of the conflict region $R^{k\ell}_{ij}(B_m)$ (cf. Fig. 3). Let c_{ijk} and $c_{i\ell j}$ be the centers of the Apollonius circles C_{ijk} and $C_{i\ell j}$.

We first determine if c_{ijk} and $c_{i\ell j}$ are in conflict with B_m. If one is in conflict but the other is not, we know the type of the conflict region. This is done using the InCircle subpredicate, which essentially computes the sign of the quantity $\delta(C, B_m)$, where C is an Apollonius circle (cf. Section 8.3). If either both endpoints of $\alpha_{ij}^{k\ell}$ are in conflict with B_m, or no endpoint of $\alpha_{ij}^{k\ell}$ is in conflict with B_m, then we compute the type of the shadow region $S_{ij}(B_m)$, by means of the ShadowRegionType subpredicate. We consider two cases:

(1) $c_{ijk}, c_{i\ell j} \in R_{ij}^{k\ell}(B_m)$. If $\tilde{S}_{ij}(B_m)$ is not of the type $(-\infty, \xi) \cup (\eta, \infty)$, then the entire edge is in conflict with B_m. Otherwise, we need up to two OrderOnBisector tests to determine if the interior of $\alpha_{ij}^{k\ell}$ is also in conflict with B_m.
(2) $c_{ijk}, c_{i\ell j} \notin R_{ij}^{k\ell}(B_m)$. If $\tilde{S}_{ij}(B_m)$ is not of the type (ξ, η), then the interior is not in conflict with B_m. Otherwise, we again need up to two OrderOnBisector tests to determine if the interior of $\alpha_{ij}^{k\ell}$ is in conflict with B_m.

In the subsequent subsections we prove the following lemma:

Lemma 8 *The algebraic degree of the* FiniteEdgeConflictType *predicate is 16.*

8.1 *The* ShadowRegionType *subpredicate*

Let B_i, B_j be sites and B_k be a query site. We show how to determine the type of $S_{ij}(B_k)$ (cf. Fig. 2). The first observation is that any finite point on the boundary of $S_{ij}(B_k)$ has to be the center of either the Apollonius circle C_{ijk} or the Apollonius circle C_{ikj}. The existence of C_{ijk} and C_{ikj} reduces to determining the number of positive roots of a quadratic equation. This is the Existence primitive discussed below and its degree is 5.

If we know that C_{ijk} exists, but C_{ikj} does not, then we know that $\tilde{S}_{ij}(B_k)$ is of the form $(-\infty, \xi)$. Similarly, if C_{ikj} exists, but C_{ijk} does not, then $\tilde{S}_{ij}(B_k)$ is of the form (η, ∞). If none of the two exist then $\tilde{S}_{ij}(B_k)$ is equal to either $\emptyset$ or $(-\infty, \infty)$. Analogously, if both tritangent Apollonius circles exist, $\tilde{S}_{ij}(B_k)$ is either of the form (ξ, η) or $(-\infty, \xi) \cup (\eta, \infty)$. In these last two cases we need an additional test to determine the type of $S_{ij}(B_k)$.

If both Apollonius circles C_{ijk} and C_{ikj} exist, then $\delta(B_k, L_{ij})\delta(B_k, L_{ji}) > 0$. Moreover, if $\delta(B_k, L_{ij}) < 0$, then $\tilde{S}_{ij}(B_k) = (-\infty, \xi) \cup (\eta, \infty)$. Hence we can distinguish between the cases $(-\infty, \xi) \cup (\eta, \infty)$ and (ξ, η) by looking at the sign of $\delta(B_k, L_{ij})$. If neither C_{ijk} nor C_{ikj} exist, then we can show that $\delta(B_k^\epsilon, L_{ij})\delta(B_k^\epsilon, L_{ji}) > 0$. In particular, if $\delta(B_k^\epsilon, L_{ij}) < 0$, then $\tilde{S}_{ij}(B_k) = (-\infty, \infty)$. Hence, we can distinguish between the cases $(-\infty, \infty)$ and $\emptyset$, by computing the sign of $\delta(B_k^\epsilon, L_{ij})$. This is the DistanceFromBitangent subpredicate and its algebraic degree is 6 (cf. Section 7.2). Notice that our perturbation is applied again, but does not affect the algebraic degree.

Lemma 9 *The algebraic degree of the* ShadowRegionType *subpredicate is 6.*

8.2 The Existence *primitive*

In this section we show how to determine whether a triple of mutually visible sites has a tritangent Apollonius circle or not. Let B_ν, $\nu = i, j, k$, be three sites. We are interested in only two kinds of tritangent Apollonius circles: the interior and exterior tritangent circles.

The existence of the tritangent Apollonius circle for B_ν, $\nu = i, j, k$, is equivalent to that for the sites B_ν^*, $\nu = i, j, k$, defined in Section 4.3. When we perform the transformation from the $\mathcal{Z}$ to the $\mathcal{Z}^*$-plane the three sites do not have any interior tritangent circles. In this context we want to see if the solution computed in Section 4.3 is indeed an exterior tritangent Apollonius circle, or whether it is a circle that contains both sites B_ν^*, $\nu = j, k$. In the first case the sought for Apollonius circle exists, whereas in the second case the answer is negative.

Let us consider the sites B_ν^*, $\nu = j, k$. The tritangent Apollonius circle of interest separates the plane into two regions, one bounded and one unbounded. If the tritangent Apollonius circle exists, then the point at infinity is on the same region of the plane with the sites for which r_ν^* is positive and on different regions with the sites for which r_ν^* is negative. On the $\mathcal{W}$-plane this can be expressed as follows. The point at infinity on the $\mathcal{Z}^*$-plane is mapped at the origin on the $\mathcal{W}$-plane. Requiring that the site B_ν^*, $\nu = j, k$, is on the unbounded side of C^* is equivalent to requiring that W_ν is on the same half-plane with respect to L as the origin, on the $\mathcal{W}$-plane (recall that the inversion transformation preserves the tangency and containment relationships). Similarly, if B_ν^* is on the bounded side of C^* on the $\mathcal{Z}^*$-plane, we require that W_ν is on different half-planes, with respect to L, with the origin. In any case, given the orientation that we chose for L, the above requirements are equivalent to requiring that the origin on the $\mathcal{W}$-plane is on the positive half-plane with respect to L, which algebraically can be written as:

$$\bar{a}_{ijk} \cdot 0 + \bar{b}_{ijk} \cdot 0 + \bar{c}_{ijk} > 0.$$

In other words we require that $\bar{c}_{ijk} > 0$. We can compute the sign of $\bar{c}_{ijk}$ in the following two ways.

One approach is to consider the explicit expression for $\bar{c}_{ijk}$ given by relation (9). This reduces to sign determination by relation (12). Its degree is 10 in the original coordinates.

However, we can do better than that. Following the analysis of Sections 7.2 and 4.3 we can write the defining equation for $\bar{c}_{ijk}$. This is a quadratic equation of the form:

$$\gamma \bar{c}_{ijk}^2 + \beta \bar{c}_{ijk} + \alpha = 0, \tag{18}$$

where

$$\gamma = (D_{jk}^{u})^2 + (D_{jk}^{v})^2, \quad \beta = -2(D_{jk}^{u} D_{jk}^{u\rho} + D_{jk}^{v} D_{jk}^{v\rho}),$$

$$\alpha = (D_{jk}^{u\rho})^2 + (D_{jk}^{v\rho})^2 - (D_{jk}^{uv})^2.$$

In the original coordinates, and given that $p_\nu^* > 0$, $\nu = j, k$, these expressions reduce to:

$$\gamma = (E_{ijk}^{xp})^2, \quad \beta = -2(E_{ijk}^{xp} E_{ijk}^{xr} + E_{ijk}^{yp} E_{ijk}^{yr}), \quad \alpha = (E_{ijk}^{xr})^2 + (E_{ijk}^{yr})^2 - (E_{ijk}^{xy})^2.$$

Note that the discriminant $\Delta_{\bar{c}_{ijk}}$ of equation (18) is $\Delta_{\bar{c}_{ijk}} = (E_{ijk}^{xy})^2\, \Gamma_{ijk}$. If E_{ijk}^{xy} is positive, then c is the largest of the two roots of (18). If E_{ijk}^{xy} is negative, then $\bar{c}_{ijk}$ is the smallest of the two roots of (18). Then,

(1) If $E_{ijk}^{xy} > 0$
  (a) If $\alpha < 0$, then $\text{sign}(\bar{c}_{ijk}) = 1$
  (b) If $\alpha \geq 0$
    (i) If $\beta > 0$, then $\text{sign}(\bar{c}_{ijk}) = 1$
    (ii) If $\beta < 0$, then $\text{sign}(\bar{c}_{ijk}) = -\text{sign}(\alpha)$
    (iii) If $\beta = 0$, then $\text{sign}(\bar{c}_{ijk}) = \text{sign}(\alpha)$
(2) If $E_{ijk}^{xy} < 0$
  (a) If $\alpha < 0$, then $\text{sign}(\bar{c}_{ijk}) = -1$
  (b) If $\alpha \geq 0$
    (i) If $\beta > 0$, then $\text{sign}(\bar{c}_{ijk}) = \text{sign}(\alpha)$
    (ii) If $\beta < 0$, then $\text{sign}(\bar{c}_{ijk}) = -1$
    (iii) If $\beta = 0$, then $\text{sign}(\bar{c}_{ijk}) = -\text{sign}(\alpha)$
(3) If $E_{ijk}^{xy} = 0$, then $\Delta_{\bar{c}_{ijk}} = 0$, which means that (18) has a double root; in this case $\text{sign}(\bar{c}_{ijk}) = \text{sign}(\beta)$.

Since the degrees of β, α and E_{ijk}^{xy}, with respect to the original coordinates, are 5, 4 and 2, respectively, we have:

Lemma 10 *The* EXISTENCE *primitive can be evaluated by determining the signs of the quantities β, α and E_{ijk}^{xy}, and it is of algebraic degree 5.*

Suppose that we are given once again four sites B_ν, $\nu = i, j, k, \ell$, and that we want to compute the sign of the distance $\delta(C_{ijk}, B_\ell)$ of B_ℓ from the tritangent Apollonius circle C_{ijk} of B_ν, $\nu = i, j, k$. We consider again the inversion transformation described in Section 4.3, but now we also transform B_ℓ to B_ℓ^* and then to W_ℓ. The problem of determining the afore-mentioned sign on the $\mathcal{Z}$-plane now reduces to determining the sign of the (signed) distance of W_ℓ from the line L. Algebraically this means that we need to compute the sign of the quantity Q_I:

$$Q_I := \bar{a}_{ijk}u_\ell + \bar{b}_{ijk}v_\ell + \bar{c}_{ijk} - \rho_\ell = \bar{a}_{ijk}(u_\ell - u_j) + \bar{b}_{ijk}(v_\ell - v_j) - (\rho_\ell - \rho_j). \quad (19)$$

Substituting the expressions for $\bar{a}_{ijk}$ and $\bar{b}_{ijk}$ from (8) we get:

$$\begin{aligned} Q_I &= \frac{D^\rho_{jk}D^u_{jk}D^u_{\ell j} + D^\rho_{jk}D^v_{jk}D^v_{\ell j} - D^\rho_{\ell j}[(D^u_{jk})^2 + (D^v_{jk})^2] \pm Z^{uv}_{jk\ell}\sqrt{\Delta_{jk}}}{(D^u_{jk})^2 + (D^v_{jk})^2} \\ &= \frac{D^u_{jk}(D^u_{\ell j}D^\rho_{jk} - D^u_{jk}D^\rho_{\ell j}) + D^v_{jk}(D^v_{\ell j}D^\rho_{jk} - D^v_{jk}D^\rho_{\ell j}) \pm Z^{uv}_{jk\ell}\sqrt{\Delta_{jk}}}{(D^u_{jk})^2 + (D^v_{jk})^2}, \end{aligned}$$

where

$$Z^{uv}_{jk\ell} = D^u_{jk}D^v_{\ell j} - D^v_{jk}D^u_{\ell j}.$$

The above expression can be greatly simplified by using the following identities:

$$D^u_{\ell j}D^\rho_{jk} - D^u_{jk}D^\rho_{\ell j} = D^{u\rho}_{jk\ell}, \qquad D^v_{\ell j}D^\rho_{jk} - D^v_{jk}D^\rho_{\ell j} = D^{v\rho}_{jk\ell},$$
$$D^u_{jk}D^v_{\ell j} - D^v_{jk}D^u_{\ell j} = D^{uv}_{jk\ell}.$$

The denominator of Q_I is strictly positive, thus the condition $Q_I \geq 0$ is equivalent to $Q_I' \geq 0$, where Q_I' is the numerator of Q_I. The expression for Q_I' in terms of the original coordinates is:

$$Q_I' = \frac{E^{xrp}_{jk\ell}E^{xp}_{ijk} + E^{yrp}_{jk\ell}E^{yp}_{ijk} + E^{xyp}_{jk\ell}\sqrt{\Gamma_{ijk}}}{(p_j^*)^2(p_k^*)^2p_\ell^*}$$

where

$$E^{stq}_{\mu\nu\lambda} = \begin{vmatrix} s_\mu^* & t_\mu^* & q_\mu^* \\ s_\nu^* & t_\nu^* & q_\nu^* \\ s_\lambda^* & t_\lambda^* & q_\lambda^* \end{vmatrix}, \qquad s, t, q \in \{x, y, r, p\}, \quad \mu, \nu, \lambda \in \{j, k, \ell\}.$$

Since $p_\nu^* > 0$, $\nu = j, k, \ell$, in order to determine the InCircle subpredicate we need to determine the sign of the quantity:

$$Q_I'' = E^{xrp}_{jk\ell}E^{xp}_{ijk} + E^{yrp}_{jk\ell}E^{yp}_{ijk} + E^{xyp}_{jk\ell}\sqrt{\Gamma_{ijk}}.$$

To do this we can use relation (12); since the degree of $E^{srp}_{jk\ell}$, $s \in \{x, y\}$, and $E^{xyp}_{jk\ell}$ is 4, the degree of E^{sp}_{ijk}, $s \in \{x, y\}$, is 3, and the degree of Γ_{ijk} is 6, we conclude that the degree of the InCircle predicate is 14.

Theorem 11 *The* InCircle *predicate can be evaluated by determining the sign of the quantity Q''_I, and it is of algebraic degree 14 in the input quantities.*

Remark: If all three circles are of the same radius, then $r^*_\nu = 0$, $i = j, k, \ell$, and the sign of Q''_I is equal to the sign of $E^{xyp}_{jk\ell}$. In this case, $E^{xyp}_{jk\ell}$ is nothing but the determinant involved in the usual InCircle test for points.

8.4 The OrderOnBisector *subpredicate*

Let B_i and B_j be two sites and let π_{ij} be their oriented bisector. Let p and q be two points on π_{ij}. In our setting, p and q are the centers of tritangent Apollonius circles, thus each defined by B_i, B_j and a third site each. The aim of this section is to discuss how to determine the order of p and q on π_{ij}. Let A_{ij} denote the line going through the centers b_i and b_j of B_i and B_j. If $r_i \neq r_j$, then π_{ij} is a branch of hyperbola and A_{ij} is the axis of symmetry of this hyperbola. If $r_i = r_j$, then π_{ij} is a line and A_{ij} is a line perpendicular to π_{ij}, going through the midpoint of the segment $b_i b_j$. We call o_{ij} the intersection of π_{ij} and A_{ij}. In both cases the bitangent Apollonius circle centered at o_{ij} is the bitangent Apollonius circle of smallest weight among all bitangent Apollonius circles of B_i and B_j (exterior bitangent Apollonius circles have positive weight equal to their radius and interior bitangent Apollonius circles have negative weight, with absolute value equal to their radius). The weight of bitangent Apollonius circles is a strictly monotone function in the two half-bisectors of π_{ij} defined with respect to o_{ij}. More precisely, let $C(p)$ denote the bitangent Apollonius circle centered at $p \in \pi_{ij}$ and let $w(p)$ denote the weight of $C(p)$. Then for all $p, q \in \pi_{ij}$ with $o_{ij} \preccurlyeq p \prec q$ we have $w(p) < w(q)$. Similarly, for all $p, q \in \pi_{ij}$ with $p \prec q \preccurlyeq o_{ij}$, we have $w(p) > w(q)$. The above observation suggests a way to determine the order of $p, q \in \pi_{ij}$. If p and q are on different sides w.r.t. A_{ij} then we know the order immediately. If both p and q are on the same side w.r.t. A_{ij}, then we can determine the order of p and q on π_{ij} by looking at the sign of $w(p) - w(q)$.

To determine the side of A_{ij} in which p and q reside we can use the Orientation predicate discussed in Section 9.1. The subpredicate corresponding to the computation of the sign of $w(p) - w(q)$ is called the RadiiDifference subpredicate and it is discussed in the following subsection.

8.5 *The* RadiiDifference *subpredicate*

In this section we show how to determine the difference of the weights of the Apollonius circles whose centers lie on a common bisector. Let B_i, B_j be two sites and let π_{ij} be their oriented bisector. Let p, q be two points on π_{ij}. We want to determine whether $p \equiv q$, $p \prec q$ or $p \succ q$. In our case, p and q are the centers of CCW-tritangent Apollonius circles, i.e., they are defined by B_i, B_j and a third site each. Without loss of generality we can assume that p is the center c_{ijk} of the CCW-tritangent Apollonius circle C_{ijk} of B_i, B_j, B_k, and that q is the center $c_{i\ell j}$ of the CCW-tritangent Apollonius circle $C_{i\ell j}$ of B_i, B_ℓ, B_j. The RadiiDifference test is performed only when we are unable to determine the ordering of the Apollonius centers c_{ijk} and $c_{i\ell j}$ on π_{ij} by one of the previous predicates or subpredicates. This implies that both C_{ijk} and $C_{i\ell j}$ exist, and moreover they are on the same half of π_{ij} with respect to o_{ij}. We call w_{ijk} and $w_{i\ell j}$ the weights of the Apollonius circles C_{ijk} and $C_{i\ell j}$ respectively. What the RadiiDifference primitive needs to compute is the sign of the difference:

$$Q_R := w_{ijk} - w_{i\ell j}.$$

As we saw in Section 4.3, the weights of the Apollonius circles C_{ijk} and $C_{i\ell j}$ are given by:

$$w_{ijk} = \frac{1}{2\bar{c}_{ijk}} - r_i, \qquad w_{i\ell j} = \frac{1}{2\bar{c}_{i\ell j}} - r_i,$$

where we used the fact that $\bar{c}_{ijk}, \bar{c}_{i\ell j} > 0$, by the Existence predicate. Hence:

$$Q_R = \frac{1}{2}\left(\frac{1}{\bar{c}_{ijk}} - \frac{1}{\bar{c}_{i\ell j}}\right).$$

In other words it suffices to compute the sign of the quantity:

$$t = \frac{1}{\bar{c}_{ijk}} - \frac{1}{\bar{c}_{i\ell j}}. \tag{20}$$

Let us consider the equations defining $\bar{c}_{ijk}$ and $\bar{c}_{i\ell j}$ (e.g., (18) is the defining equation of $\bar{c}_{ijk}$). These are quadratic equations of the form:

$$\gamma_\tau y_\tau^2 - 2\beta_\tau y_\tau + \alpha_\tau = 0, \qquad \tau = 1, 2, \tag{21}$$

where $\bar{c}_{ijk}$ is the one of the roots of the equation for $\tau = 1$, and $\bar{c}_{i\ell j}$ is one of the roots of the equation for $\tau = 2$. The algebraic degrees of γ_τ, β_τ and α_τ, are 6, 5, and 4, respectively. We are going to rewrite the equations (21) in such a way, so that the unknown is not $\bar{c}_{ijk}$ or $\bar{c}_{i\ell j}$, but rather $1/\bar{c}_{ijk}$ and $1/\bar{c}_{i\ell j}$. The equations then become:

$$f_\tau := \alpha_\tau \frac{1}{y_\tau^2} - 2\beta_\tau \frac{1}{y_\tau} + \gamma_\tau = 0, \qquad \tau = 1, 2. \tag{22}$$

or

$$f_\tau = \alpha_\tau x_\tau^2 - 2\beta_\tau x_\tau + \gamma_\tau = 0, \qquad \tau = 1, 2. \tag{23}$$

In what follows we are going to assume that $\alpha_\tau > 0$, $\tau = 1, 2$. The case $\alpha_1\alpha_2 = 0$ will be discussed in detail in Section 8.5.4.

8.5.1 Straightforward methods of evaluation

We first make the observation that we know which one of the roots of equations (21) to pick. We know that the root that we want is the one given by (9): If $E_{ijk}^{xy} > 0$ we are interested in the largest root of (21; $\tau = 1$). If $E_{ijk}^{xy} < 0$ we are interested in the smallest root of (21; $\tau = 1$). Finally, if $E_{ijk}^{xy} = 0$, we are interested in the double root of (21; $\tau = 1$). The analysis for $\bar{c}_{i\ell j}$ is entirely analogous. Once we know which of the roots of (21) is of interest, we can determine which of the roots of (23) is of interest. For example, suppose that we are interested in the largest root of (21). Then, if $\gamma_\tau > 0$ we are interested in the smallest root of (23), whereas if $\gamma_\tau < 0$ we are interested in the largest root of (23). The argumentation is symmetric when we are interested in the smallest root of (21). Throughout the rest of this section we shall concentrate on the comparison of the largest roots of equations (23). The analysis for the remaining three cases is similar.

One straightforward approach is to evaluate the sign of t by substituting in (20) the expressions for $1/\bar{c}_{ijk}$ and $1/\bar{c}_{i\ell j}$. Then we get (using the notation from Section 4):

$$t = \frac{\beta_1 + \sqrt{\Delta_1}}{\alpha_1} - \frac{\beta_2 + \sqrt{\Delta_2}}{\alpha_2} = \frac{-J + \alpha_2\sqrt{\Delta_1} - \alpha_1\sqrt{\Delta_2}}{\alpha_1\alpha_2}. \tag{24}$$

Since $\alpha_1\alpha_2 > 0$ it suffices to compute the sign of the numerator of t. This is a quantity of the form $X_0 + X_1\sqrt{Y_1} + X_2\sqrt{Y_2}$, where, in general, $Y_1 \neq Y_2$ and $Y_1, Y_2 > 0$. The degrees of X_0, X_1, X_2, Y_1 and Y_2 are 11, 8, 8, 6 and 6, respectively. Following the analysis of Section 6, in order to determine the sign of $X_0 + X_1\sqrt{Y_1} + X_2\sqrt{Y_2}$, we need to determine, in the worst case the sign of the quantity $(X_0^2 + X_1^2Y_1 - X_2^2Y_2)^2 - 4X_0^2X_1^2Y_2$, which is a degree 36 quantity in the input.

Let us consider the evaluation procedure for comparing the two larger roots. The same problem is solved, albeit by other methods, by the procedures detailed later in Figures 9 and 10. Here, the evaluation starts by computing J and E, in the notation of the above figures, or of Section 4. If their signs are the same, then one needs to compute $L_1 := J^2 - E - 2\alpha_1^2\Delta_2$. When $L_1 > 0$, the algorithm tests $L_2 := L_1^2 - \alpha_1^2\alpha_2^2\Delta_1\Delta_2$. The tests described so far define a tree whose branches depend on the different signs. If all branches are assumed equally likely, it is straightforward to compute the probability that we visit a specific node.

We wish to bound the expected total bit complexity of the procedure, assuming all operations have complexity linear in the bit size of the operands. Let us suppose that the input parameters have unit bit size. Then, in order to bound the bit cost per node, we multiply the number of operations by the degree of the computed quantity at every step. The degree in the input data of J, E, L_1, L_2 are 9, 18, 18, and 36, respectively, whereas the number of arithmetic operations to compute each one from the previously computed quantities is 3, 5, 4, and 3. This gives $27 + 90 + 72/2 + 108/4 = 180$; for instance, L_1 is required with probability $1/2$ hence its overall cost of $18 \cdot 4 = 72$ is divided by 2. We shall see later that this is significantly higher than the respective estimates with other methods.

We are now going to consider a slightly different approach. Above what we did was to substitute in (20) the values of $1/\bar{c}_{ijk}$ and $1/\bar{c}_{i\ell j}$ directly. Now we are only going to substitute the value of $1/\bar{c}_{i\ell j}$ directly. This will yield a quadratic equation in terms of t that we will need to analyze. Indeed, substituting $1/\bar{c}_{ijk}$ in terms of t and $1/\bar{c}_{i\ell j}$ in (23; $\tau = 1$), we get:

$$\alpha_1 \left(t + \frac{1}{\bar{c}_{i\ell j}} \right)^2 - 2\beta_1 \left(t + \frac{1}{\bar{c}_{i\ell j}} \right) + \gamma_1 = 0,$$

which can be rewritten as a quadratic polynomial in terms of t:

$$\bar{\alpha}_1 t^2 + \bar{\beta}_1 t + \bar{\gamma}_1 = 0, \tag{25}$$

where:

$$\bar{\alpha}_1 = \alpha_1, \qquad \bar{\beta}_1 = 2\alpha_1 \frac{1}{\bar{c}_{i\ell j}} - 2\beta_1 = f_1'(\frac{1}{\bar{c}_{i\ell j}}),$$
$$\bar{\gamma}_1 = \alpha_1 \frac{1}{(\bar{c}_{i\ell j})^2} - 2\beta_1 \frac{1}{\bar{c}_{i\ell j}} + \gamma_1 = f_1(\frac{1}{\bar{c}_{i\ell j}}).$$

The problem of determining the sign of t now reduces to determining the sign of the appropriate root of (25). This can be done by using Descartes' rule on (25), which calls for computing the signs of $\bar{\beta}_1$ and $\bar{\gamma}_1$. Here $\bar{c}_{i\ell j}$ is the appropriate root of (21; $\tau = 2$), such that $1/\bar{c}_{i\ell j}$ is the largest root of (23; $\tau = 2$). Both $\bar{\beta}_1$ and $\bar{\gamma}_1$ are expressions of the form $(X_0 + X_1\sqrt{Y})/Z$, where $Z > 0$, and their signs can be evaluated using relation (12). The degrees of X_0, X_1 and Y are 9, 6 and 6, respectively, for $\bar{\beta}_1$ and 14, 11 and 6, respectively for $\bar{\gamma}_1$. Hence the highest algebraic degree involved in the evaluation of the signs of the roots of (25) is 28.

The discussion of Section 4 can be directly applied to the RadiiDifference predicate, hence offering two means of handling this predicate, either by resultants of 3 bivariate polynomials or by Sturm sequences. Both approaches yield the same maximum degree, although the former requires the additional quantity E, which raises this degree. Moreover, the rough comparison of precision and bit operations indicates that Sturm sequences are more efficient for

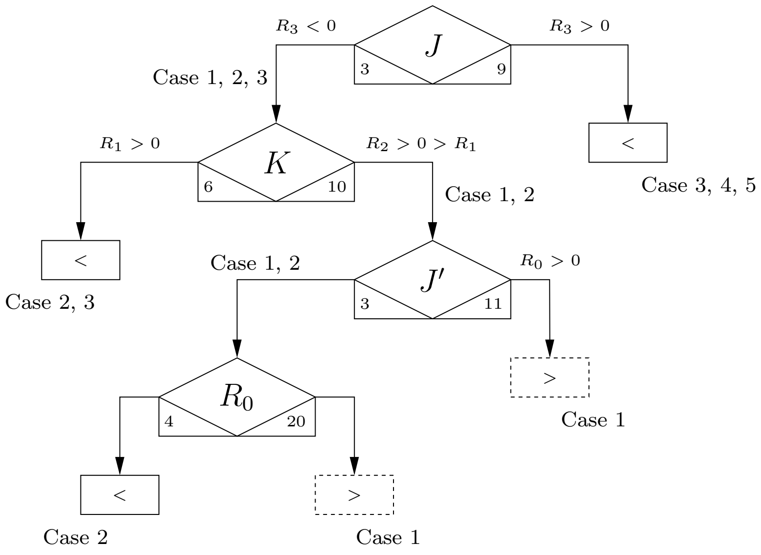


Fig. 8. Evaluation procedure for $x_1^- \diamond x_2^+$, for $\diamond \in \{<, >\}$.

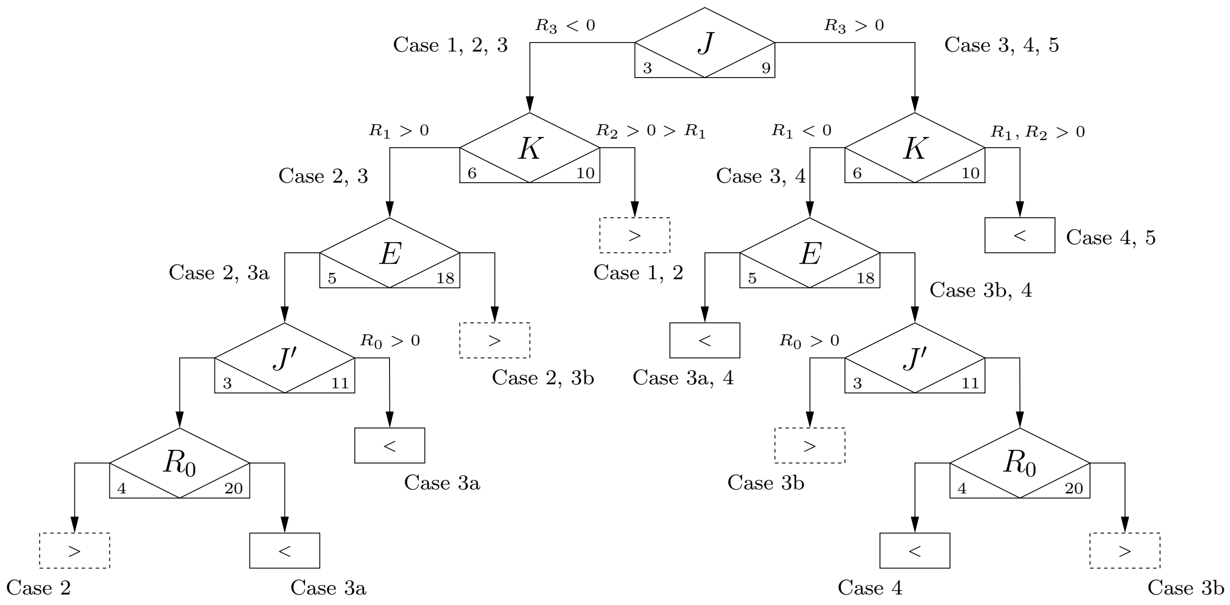


Fig. 9. Evaluation procedure for $x_1^+ \diamond x_2^+$, for $\diamond \in \{<, >\}$, using the resultant.

the predicate in hand. In the two paragraphs that follow we use the definitions and notations of Section 4.

8.5.2 Evaluation by resultants

The resultant R corresponding to the RadiiDifference subpredicate is $R = \det M$, where M is the matrix of Example 1. Using expression (1), the maximum degree of the coefficients of t^3, t^2, t, 1, are, respectively, 9, 18, 10, 20, in the input quantities. The coefficient of t^4 is always positive. Applying the analysis of Section 4.1, one may follow the evaluation procedure of Figure 8 when comparing a small and a large root, or that of Figure 9 for comparing the two large roots.

In these figures, the bottom part of each evaluation shows the number of operations needed to compute the quantity, given the previously computed ones, and its degree in the input data. In counting operations, we ignore negations. Given the coefficients of the f_i and the discriminants Δ_i, it is possible to compute the quantities in Figure 9 with an expected number of operations of $3 + 6 + 5/2 + 3/4 + 4/8 = 12\frac{3}{4}$, assuming each branch is equally likely. A more realistic measure is to simulate bit complexity cost by multiplying the number of operations at each node by the corresponding degree in the input data. Then, the same predicate costs $27 + 60 + 90/2 + 33/4 + 80/8 = 150\frac{1}{4}$. Another way of measuring expected complexity is by considering that all 6 cases are equally likely. To identify cases 1 and 5, 9 operations are required. For cases 2 and 4, we need $3 + 6 + 5/2 + 3/4 + 4/8 = 12\frac{3}{4}$, whereas for each of cases 3a and 3b we need $3 + 6 + 5 + 3/2 + 4/4 = 16\frac{1}{2}$ operations on the average. This gives an overall average of $12\frac{3}{4}$ operations.

8.5.3 Evaluation by Sturm sequences

Let us consider the evaluation of the Sturm sequence $(P_i)_i$ at $p = \frac{\beta_i}{\alpha_i} \in \mathbb{R}$. The first sign depends on α_1 because it shows whether the parabola is facing upwards or downwards.

sign$(P_i(p))$		
i	assuming $\Delta_1, \Delta_2 > 0$	assuming $\Delta_1, \Delta_2, \alpha_1, \alpha_2 > 0$
0	$-\mathrm{sign}(\alpha_1)$	$-$
1	0	0
2	$\mathrm{sign}(\alpha_1)$	$+$
3	$\mathrm{sign}(J)$	$\mathrm{sign}(J)$
4	$\mathrm{sign}(\alpha_1)\mathrm{sign}((\alpha_1 K + 2\alpha_2\Delta_1)^2)\mathrm{sign}(4JJ' - G^2)$	$\mathrm{sign}((\alpha_1 K + 2\alpha_2\Delta_1)^2)\mathrm{sign}(4JJ' - G^2)$

If $\Delta_\tau > 0$ and $\alpha_\tau > 0$, then the degree of the tested quantities in the input data is 0 for P_0, P_1, P_2, 9 for $P_3(p)$, and 20 for P_4, because the degree of α_τ, β_τ and γ_τ are, respectively, 4, 5, 6. For $q = \infty$, we have the following signs:

sign$(P_i(\infty))$		
i	assuming $\Delta_1, \Delta_2 > 0$	assuming $\Delta_1, \Delta_2, \alpha_1, \alpha_2 > 0$
0	$\mathrm{sign}(\alpha_1)$	$+$
1	$\mathrm{sign}(\alpha_1)\mathrm{sign}(\alpha_2)$	$+$
2	$-\mathrm{sign}(\alpha_1)$	$-$
3	$-\mathrm{sign}(\alpha_1)\mathrm{sign}(\alpha_1 K + 2\alpha_2\Delta_1)$	$-\mathrm{sign}(\alpha_1 K + 2\alpha_2\Delta_1)$
4	$\mathrm{sign}(\alpha_1)\mathrm{sign}((\alpha_1 K + 2\alpha_2\Delta_1)^2)\mathrm{sign}(4JJ' - G^2)$	$\mathrm{sign}((\alpha_1 K + 2\alpha_2\Delta_1)^2)\mathrm{sign}(4JJ' - G^2)$

Looking into the second column again, we have degree 0 for P_0, P_1, P_2 and 14 for $P_3(\infty)$. The identity $\alpha_1 K + 2\alpha_2\Delta_1 = -2\beta_1 J + \alpha_1 G$ offers an alternative means of determining $\text{sign}(P_3(\infty))$. Since the two polynomials have no common root, computing $\text{sign}(f_2(x_1^+)) = V_P(p) - V_P(\infty)$ yields an answer in $\{-1, 1\}$. The same Sturm sequence may be evaluated at $-\infty$ and p to yield $\text{sign}(f_2(x_1^-)) = V_P(-\infty) - V_P(p) \in \{-1, 1\}$. At $x \mapsto p$ the signs are the same as above, and at $-\infty$ they are

$$+, +, -, -\text{sign}(P_3(\infty)), \text{sign}(P_4),$$

because the degrees of the polynomials $P_0(x), \dots, P_4(x)$ are 2, 3, 2, 1, 0 in x. We can analogously evaluate the Sturm sequence $(Q_i)_i$ at $p = \frac{\beta_i}{\alpha_i}$ and $q = \infty$. Again we use the fact that $\Delta_\tau > 0$ and in the last column we further assume that $\alpha_\tau > 0$:

$\text{sign}(Q_i(p))$		
i	assuming $\Delta_1, \Delta_2 > 0$	assuming $\Delta_1, \Delta_2, \alpha_1, \alpha_2 > 0$
0	$-\text{sign}(\alpha_1)$	$-$
1	0	0
2	$\text{sign}(\alpha_1)$	$+$
3	$\text{sign}(\alpha_2)\text{sign}(J^2)\text{sign}(\alpha_1\Delta_2 + \alpha_2 K)$	$\text{sign}(J^2)\text{sign}(\alpha_1\Delta_2 + \alpha_2 K)$

For $q = \infty$:

$\text{sign}(Q_i(\infty))$		
i	assuming $\Delta_1, \Delta_2 > 0$	assuming $\Delta_1, \Delta_2, \alpha_1, \alpha_2 > 0$
0	$\text{sign}(\alpha_1)$	$+$
1	$\text{sign}(\alpha_1)\text{sign}(\alpha_2)$	$+$
2	$-\text{sign}(\alpha_2)\text{sign}(J)$	$-\text{sign}(J)$
3	$\text{sign}(\alpha_2)\text{sign}(J^2)\text{sign}(\alpha_1\Delta_2 + \alpha_2 K)$	$\text{sign}(J^2)\text{sign}(\alpha_1\Delta_2 + \alpha_2 K)$

Evaluating this Sturm sequence at $-\infty$, we get the following sign sequence:

$$+, +, \text{sign}(J), \text{sign}(Q_3)$$

because the degrees of the polynomials $Q_0(x), \dots, Q_3(x)$ are 2, 2, 1, 0 in x. Figure 10 shows the evaluation of the predicate using Sturm sequences, when we need to compare the two larger roots x_τ^+. This is to be juxtaposed to Figure 9. The shown procedure can also handle the cases where any or several of the tested quantities vanish, but this is not made explicit in Figure 10 for the sake of simplicity and readability.

If all branches are equally likely, the expected number of operations is $3 + 6 + 4/2 + 3/4 + 4/8 = 11\frac{1}{4}$. If each number of operations is multiplied by the corresponding degree in the input data, this count becomes $27 + 60 + 56/2 + 33/4 + 80/8 = 133\frac{1}{4}$. If we consider all 6 cases to be equally likely, cases 1 and

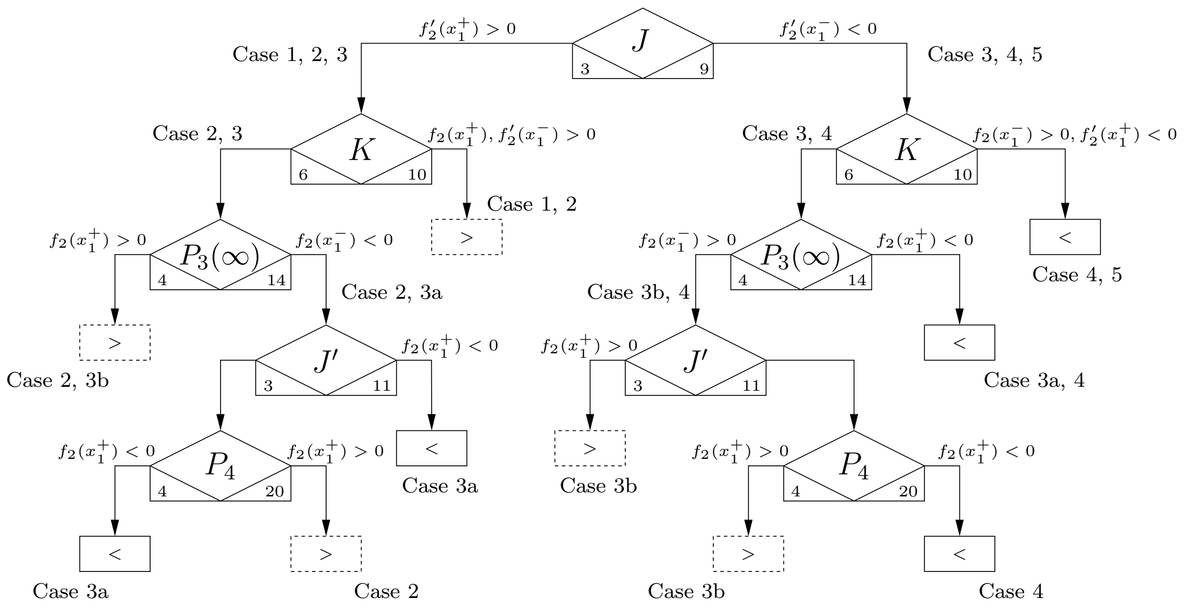


Fig. 10. Evaluation procedure for deciding $x_1^+ \diamond x_2^+$, where $\diamond \in \{<, >\}$. At each tested quantity we indicate the number of operations needed in order to compute it (left) and its algebraic degree in the input (right).

5 require $3 + 6 = 9$ operations, cases 2 and 4 require $3 + 6 + 4/2 + 3/4 + 4/8 = 12\frac{1}{4}$ operations and cases 3a and 3b require $3 + 6 + 4 + 3/2 + 4/4 = 15\frac{1}{2}$ operations. This gives an overall average of $12\frac{1}{4}$ operations. All are lower than the respective numbers of operations using resultants. The evaluation tree in Figure 10 is not the only possible one. Its design, however, reflects our main aims:

(1) Conclude as fast as possible (e.g., Case 5 is decided after only two comparisons, those of J and K). This is also achieved by using certain quantities as filters for others (e.g., J' is a filter for P_4).
(2) Re-use computed quantities (e.g., J can be used to compute $P_3(\infty)$ and P_4; K or J' can be used to compute P_4). This helps minimizing the expected number of operations in the tree.
(3) Compute high-degree quantities as rarely as possible, i.e., the high-degree quantities should appear as low as possible in the tree (e.g., P_4, the highest-degree quantity, is a leaf of the tree).

Analogous procedures are obtained for comparing the other root pairs, e.g., Fig. 8 shows the procedure for $x_1^- \diamond x_2^+$, $\diamond \in \{<, >\}$.

Now, a further reduction in the degree of the tested polynomials is possible if we write them in terms of the input quantities by completely developing all intermediate quantities $(\alpha_\tau, \beta_\tau, \gamma_\tau)$ as functions of the input parameters. Then P_4 factorizes to two polynomials of degrees 12 and 8, respectively. To understand their complexity, consider that the former has 205 monomials in the input quantities. However, the same effect is not possible with the resultant method, because it requires testing E which does not factorize as a polynomial in the input quantities, hence it gives a test of degree 18. Due to the factorization of P_4 the bottlenecks of our method become the degenerate cases $\alpha_1 \neq 0$, $\alpha_2 = 0$ or $\alpha_1 = 0$, $\alpha_2 \neq 0$, in which case we need quantities of degree 16 to answer our problem. We discuss these cases in the next subsection.

Lemma 12 *There is an algorithm for deciding* RadiiDifference *that reduces the predicate to testing the sign of the difference of two specific roots of quadratic equations. This algorithm tests quantities of degree at most 16 in the input coefficients by using Sturm sequences as described above.*

8.5.4 Sturm sequence degeneracies

Due to the factorization of P_4 to two expressions of degree 12 and 8 in the input parameters respectively, the maximum algebraic degree now appears in the case of degenerate input, namely $\alpha_1\alpha_2 = 0$, $|\alpha_1| + |\alpha_2| > 0$. Geometrically this corresponds to 3 sites which have a common tangent line, or equivalently to 3 sites whose Apollonius circle has infinite radius. In this case we need quantities of degree 16 to answer our problem. It is interesting to note that these quantities do not factorize to expressions of lower degree. Recall that factorization does not commute with taking projections of polynomials modulo an ideal: here the ideal is defined by the polynomial α_τ, $\tau = 1, 2$, developed in terms of the input parameters. In particular, the Sturm sequence P of $f_1, f_1' f_2$, assuming $\alpha_1 = 0 < \alpha_2$, ends with $P_3(x) = 4\beta_1^2\alpha_2 f_2(\gamma_1/(2\beta_1))$. The quantity $4\beta_1^2 f_2(\gamma_1/(2\beta_1))$ is of degree 16. Notice that the Sturm sequences cannot be obtained simply by specializing the quantities that vanish because specialization does not commute with pseudo-remaindering.

One may consider as degenerate any configuration that makes one or more of the tested quantities equal to zero. One such degenerate setting is when $\alpha_\tau > 0$, $\tau = 1, 2$, but $2\beta_1 J - \alpha_1 G = 0$, in which case the polynomial $P_3(x)$ is a constant. In this case we have:

$$2\beta_1 J = \alpha_1 G \quad \Longleftrightarrow \quad (x_1^+ - x_2^-)(x_2^+ - x_1^+) + (x_1^- - x_2^-)(x_2^+ - x_1^-) = 0.$$

In this degenerate case, we may compute the Sturm sequence under this hypothesis and see that all tested quantities are of lower degree. In particular, P_3 becomes $2\alpha_1\Delta_1 J'$, hence the maximum degree of any tested quantity is 11. Note that this expression for P_3 can also be obtained by simply specializing the polynomial $P_3(x)$.

We have checked that all degeneracies can be handled by the procedure of Figure 10 (and the analogous procedures for testing different pairs of roots) without modifying the shown tree substantially. More specifically, for certain nodes, the zero case can be incorporated in one of the two non-zero cases considered already. For the rest of the nodes, when the tested quantity vanishes, it is possible to conclude almost immediately and decide which case occurs. Therefore, the maximum degree of expressions tested by our algorithm becomes 16 when dealing with arbitrary inputs, including the degenerate cases. This discussion, together with that of the previous section, proves our main algorithmic result.

Theorem 13 *It is possible to implement the algorithm of [9] for constructing the Apollonius diagram by testing quantities of degree at most 16 in the input parameters.*

Note that, in order to compute the RadiiDifference primitive, it is also possible to use the polynomials $f_\tau(y)$, $\tau = 1, 2$, where $y = 1/x$; cf. also the definition of f_τ in (21). The maximum algebraic degree of any tested quantity in this case is 16. On the upside, we can avoid all of the above degeneracies, since $\gamma_\tau > 0$, $\tau = 1, 2$, by definition. This yields an alternative approach with the same maximum degree as that of Theorem 13.

9 The last two predicates

9.1 The Orientation *predicate*

In this section we deal with the usual orientation predicate when one of the points is the center of a tritangent Apollonius circle. Let $B_\nu = \{(x_\nu, y_\nu), r_\nu\}$, $\nu = i, j, k$, be the defining sites, and let (c_x, c_y) be the coordinates of the center c_{ijk} of the Apollonius circle C_{ijk}. Finally, let (x_ν, y_ν), $\nu = \ell, m$ be the remaining two points involved in the Orientation predicate. We want to compute the sign of the quantity Q_O below. Following relation (7), we get:

$$Q_O = \begin{vmatrix} c_x & c_y & 1 \\ x_\ell & y_\ell & 1 \\ x_m & y_m & 1 \end{vmatrix} = \frac{1}{2\bar{c}_{ijk}} \left[- \begin{vmatrix} \bar{a}_{ijk} & \bar{b}_{ijk} & 0 \\ x_\ell & y_\ell & 1 \\ x_m & y_m & 1 \end{vmatrix} + 2\bar{c}_{ijk} \begin{vmatrix} x_i & y_i & 1 \\ x_\ell & y_\ell & 1 \\ x_m & y_m & 1 \end{vmatrix} \right].$$

The Orientation predicate is used to find the first conflict of a new site with respect to the Apollonius edges of its nearest neighbor. In this context, C_{ijk} always exists, thus $\bar{c}_{ijk} > 0$ by Section 8.2. Therefore, it suffices to determine the sign of the quantity Q'_O, which is the expression inside the square brackets above. Substituting the expressions for $\bar{a}_{ijk}$, $\bar{b}_{ijk}$ and $\bar{c}_{ijk}$ from (8), (9), we get:

$$Q'_O = \frac{Q''_O}{(p_j^*)^2(p_k^*)^2[(E_{ijk}^{xp})^2 + (E_{ijk}^{yp})^2]},$$

where

$$\begin{aligned} Q''_O = & 2(E_{ijk}^{xp}E_{ijk}^{xr} + E_{ijk}^{yp}E_{ijk}^{yr})E_{i\ell m}^{xy} + [E_{ijk}^{yp}(x_\ell^* - x_m^*) - E_{ijk}^{xp}(y_\ell^* - y_m^*)]E_{ijk}^{rp} \\ & + [2E_{ijk}^{xy}E_{i\ell m}^{xy} - E_{ijk}^{xp}(x_\ell^* - x_m^*) - E_{ijk}^{yp}(y_\ell^* - y_m^*)]\sqrt{\Gamma_{ijk}}. \end{aligned}$$

and $x_\nu^* = x_\nu - x_i$, $y_\nu^* = y_\nu - y_i$, $\nu = \ell, m$. Recall also (cf. rel. (3)) that $p_\lambda^* = (x_\lambda^*)^2 + (y_\lambda^*)^2 - (r_\lambda^*)^2$, $\lambda = j, k$. The sign of Q''_O can be computed using relation (12). The algebraic degrees of x_ν^*, y_ν^*, $\nu = \ell, m$, p_λ^*, $\lambda = j, k$, are 1, 1 and 2 respectively, whereas the degrees of E_{ijk}^{sr}, $s \in \{x, y\}$, E_{ijk}^{sp}, $s \in \{x, y, r\}$, E_{ijk}^{st}, $s, t \in \{x, y, r\}$, $E_{i\ell m}^{xy}$ and Γ_{ijk} are 2, 3, 2, 2 and 6, respectively. Hence:

Lemma 14 *The* ORIENTATION *predicate can be evaluated by determining the sign of the quantity Q''_O, and it is of algebraic degree 14.*

9.2 The ISDEGENERATEEDGE *predicate*

The predicate can be evaluated as follows. Given an Apollonius edge $\alpha_{ij}^{k\ell}$, first determine if B_ℓ touches the Apollonius circle C_{ijk}; if this is not the case then $\alpha_{ij}^{k\ell}$ is not degenerate. The next test that we have to perform is whether B_k touches the Apollonius circle $C_{i\ell j}$. Again, if this is not the case the edge $\alpha_{ij}^{k\ell}$ is not degenerate. Note that both of the tests above can be answered using the INCIRCLE subpredicate. There is still one final test to be done. We need to determine if the points c_{ijk} and $c_{i\ell j}$ are the same; $\alpha_{ij}^{k\ell}$ is degenerate if and only if c_{ijk} and $c_{i\ell j}$ coincide. The answer to this last test can be given using the ORDERONBISECTOR subpredicate.

It is worth noticing that we could use the ORDERONBISECTOR subpredicate directly, in order to resolve the ISDEGENERATEEDGE predicate. However, we choose to perform first the two INCIRCLE tests because they filter the vast majority of the cases where the answer to the ISDEGENERATEEDGE predicate is false, and moreover it has lower algebraic degree than the ORDERONBISECTOR subpredicate.

10 Experimental results

In this section we briefly describe the input data sets, and two sets of experiments, one for the overall algorithm and one focusing on the predicates.

We start with comments that apply to both sets of experiments. Our code is written in C++ and has recently become part of CGAL 3.0. All experiments were conducted on a Pentium-III architecture at 1 GHz. We used version 2.4 of CGAL, and version 4.3 and 4.2 of LEDA for the two respective sets of experiments. The compiler used was the GNU `g++` compiler, version 2.95.3 (with options `-O2 -mcpu=pentiumpro -march=pentiumpro`). We have implemented caching of intermediate expressions when evaluating the predicates, which reduces significantly the number of operations.

We consider various number types: LEDA reals, the multi precision floating point number `MP_Float` provided by CGAL, and the GNU multiprecision integer GMP [28]. We use the keywords `real`, `mpfloat` and `gmp` to refer to these number types, respectively. We moreover consider filtered versions of the above exact number types, where the filtering is dynamic and is performed via the interval arithmetic package of CGAL [29]. Other potential number types include CORE's `Expr` (cf. [30]) and GMP's multi precision rationals, but both these number types had not been fully interfaced with CGAL at the time our experiments were made. The built-in `double` of C++ is used primarily for reference, since it is inexact and, in general, may incorrectly evaluate the predicates. In particular, the SQRT method with `double` arithmetic never produced correct results for the ONLINE examples below. In the second subsection, `double` arithmetic is applied successfully to random data, but for the almost degenerate data it is insufficient, i.e., the predicates cannot be evaluated correctly.

10.1 Overall algorithm

We use two methods for evaluating the predicates. The first requires that signs of expressions involving the operations $\{+, -, \times, /, \sqrt{\ }\}$ are performed exactly. The second requires that signs of polynomial expressions are performed exactly. We refer to the two evaluation methods using the keywords SQRT and POLYNOMIAL, respectively.

We report on three series of experiments. Every series corresponds to a different type of input data. The first one consists of circles whose coordinates are random integers of bit size $b \in \{10, 20, 30, 40, 50\}$. Their radii are random non-negative integers of bit size $b' \in \{5, 10, 20, 30, 40\}$. The bit sizes of the radii were chosen to be smaller than the bit sizes of the coordinates in order to have few hidden sites. Larger bit sizes for the radii would produce many more hidden sites. The second example consists of circles whose centers lie on the parabola $y = x^2$ and whose radii are equal to their y-coordinate. The x-coordinate of these circles is a random integer of bit size $b \in \{5, 10, 15, 20, 25\}$. All circles of this data set are equidistant from the point $(0, \frac{1}{4})$ and are all

	b_{max}			10	20	30	40	50
	n	Method	Number type	$T(n,h)$ *[seconds]*				
InSquare	10^4	h		*6503*	*9994*	*9990*	*9993*	*9994*
		Sqrt	`double`	0.62	0.71	0.7	0.69	0.72
			`real`	12.47	13.95	14.64	14.55	14.86
			filter + `real`	2.48	2.86	2.84	2.84	2.81
		Polyn.	`double`	0.58	0.6	0.7	0.7	0.71
			`real`	10.45	13.15	15.07	15.29	16.11
			`gmp`	41.45	46.27	47.9	50.04	52.31
			`mpfloat`	22.57	26.58	31.18	36.59	37.52
			filter + `real`	2.27	2.42	2.3	2.42	2.64
			filter + `gmp`	2.18	2.51	2.53	2.49	2.61
			filter + `mpfloat`	2.31	2.52	2.5	2.51	2.6
	10^5	h		*30399*	*99376*	*99400*	*99345*	*99390*
		Sqrt	`double`	4.97	8.19	8.14	8.21	8.23
			`real`	93.47	147.65	153.61	152.97	153.29
			filter + `real`	18.48	30.61	30.04	30.07	29.79
		Polyn.	`double`	4.65	7.89	8.06	7.95	8.03
			`real`	78.98	141.78	154.62	161.41	162.31
			`gmp`	314.62	501.48	509.68	535.08	541.15
			`mpfloat`	162.47	280.15	321.77	383.04	381.23
			filter + `real`	15.9	27.15	27.29	27.18	27.31
			filter + `gmp`	16.06	27.31	27.41	27.43	27.59
			filter + `mpfloat`	16.18	27.22	27.16	27.22	27.39
	10^6	h		*139146*	*941875*	*941589*	*941746*	*941628*
		Sqrt	`double`	n/a	99.8	99.35	100.38	99.85
			`real`	692.06	1519.69	1609.7	1617.7	1609.05
			filter + `real`	141.78	336.1	336.05	342.35	334.55
		Polyn.	`double`	40.17	103.65	98.14	100.29	98.25
			`real`	605.77	1472.11	1651.06	1738.16	1731.99
			`gmp`	2250.5	5381.66	5474.15	5732.83	5852.29
			`mpfloat`	1142.1	2956.96	3364.22	3974.87	3982.74
			filter + `real`	113.68	302.58	303.02	306.84	302.38
			filter + `gmp`	115.95	304.12	302.44	303.61	301.9
			filter + `mpfloat`	119.03	301.98	302.27	303.67	302.27

Table 4
Experimental results for the InSquare data set. $T(n,h)$ is the total running time; n is the number of input sites and h the number of visible sites. b_{max} is the maximum bit size of the input; "n/a" implies that the algorithm could not compute a correct diagram due to numerical errors.

tangent to the x-axis. Our third data set consists of circles whose centers lie on the positive x-axis and whose radii are one-half the x-coordinate of their center. All circles of this data set are tangent to the lines $y = \pm\frac{1}{\sqrt{3}}x$. The x-coordinate of these circles is a non-negative random integer of bit size $b \in \{10, 20, 30, 40, 50\}$. We use the keywords InSquare, OnParabola and

	b_{max}			10	20	30	40	50
	n	Method	Number type	$T(n,h)$ *[seconds]*				
OnParabola	10^4	h		*63*	*2028*	*9269*	*9980*	*10000*
		Sqrt	`double`	0.07	n/a			
			`real`	5.19	8.87	22.7	90.11	304.78
			filter + `real`	0.19	1.13	30.05	92.65	345.47
		Polyn.	`double`	0.06	n/a			
			`real`	4.8	9.51	29.38	60.82	212.96
			`gmp`	4.98	17.84	32.35	37.77	49.62
			`mpfloat`	2.09	7.97	16.14	20.55	29.75
			filter + `real`	0.16	1.18	38.31	69	315.64
			filter + `gmp`	0.16	1.3	37.45	44.41	261.23
			filter + `mpfloat`	0.15	1.01	19.18	24.01	127.39
	10^5	h		*63*	*2047*	*51262*	*97636*	*99939*
		Sqrt	`double`	0.64	n/a			
			`real`	49.71	73.62	190.66	1159.98	3261.06
			filter + `real`	1.67	6.26	144.72	1237.5	3715.83
		Polyn.	`double`	0.66	n/a			
			`real`	47.8	74.54	254.88	541.77	2178.91
			`gmp`	50.07	152.83	465.87	424.45	491.72
			`mpfloat`	21.23	66.67	224.36	226.99	287.42
			filter + `real`	1.58	5.99	191.22	621.37	3365.68
			filter + `gmp`	1.81	6.44	181.89	515.53	2601.15
			filter + `mpfloat`	1.69	5.87	102.29	252.36	1402.28
	10^6	h		*63*	*2047*	*65535*	*795149*	*992546*
		Sqrt	`double`	6.17	n/a			
			`real`	509.6	716.53	1278.03	14935	34324.5
			filter + `real`	17.86	56.55	314.17	15606.8	39340.4
		Polyn.	`double`	6.81	n/a			
			`real`	485.7	732.78	1532.15	4737.4	22363.6
			`gmp`	499.98	1497.29	3679.23	5014.94	5106.87
			`mpfloat`	211.69	650.93	1762.05	2629.74	2922.99
			filter + `real`	17.12	53.27	359.56	5058.78	34832.9
			filter + `gmp`	17.22	55.01	377.02	4130.94	27359.5
			filter + `mpfloat`	17.49	53.89	254.03	2224.78	14175.7

Table 5
Experimental results for the OnParabola data set. $T(n,h)$ is the total running time; n is the number of input sites and h the number of visible sites. b_{max} is the maximum bit size of the input; "n/a" implies that the algorithm could not compute a correct diagram due to numerical errors.

OnLine, respectively. Clearly, the last two are highly degenerate, chosen so in order to illustrate the robustness of our implementation. In any case, we denote by b_{max} the maximum bit size of the input. Lastly, for each type of data and each bit size, the number n of input circles takes the values 10^4, 10^5 and 10^6.

	b_{max}			10	20	30	40	50
	n	Method	Number type	$T(n,h)$ [*seconds*]				
OnLine	10^4	h		*512*	*9892*	*9999*	*10000*	*10000*
		Sqrt	`double`	n/a				
			`real`	17.56	61.66	70.25	72.59	84.99
			filter + `real`	2.06	2.39	68.44	74.22	89.43
		Polyn.	`double`	0.5	0.56	0.76	n/a	
			`real`	18.77	68.06	75.08	87.11	84.86
			`gmp`	39.28	36.69	42.84	60.47	43.16
			`mpfloat`	16.2	14.64	17.31	25.23	17.87
			filter + `real`	1.5	2	78.16	88.23	94.8
			filter + `gmp`	1.52	2.05	65.42	73.49	69.8
			filter + `mpfloat`	1.53	1.95	31.44	34.14	32.96
	10^5	h		*512*	*91100*	*99991*	*99998*	*99996*
		Sqrt	`double`	n/a				
			`real`	161.87	573.31	692	687.52	843.15
			filter + `real`	19.89	22.67	555.94	714.3	857.98
		Polyn.	`double`	5.47	7.25	n/a		
			`real`	167.24	643.88	773.17	831.47	851.86
			`gmp`	401.82	383.24	466.34	573.5	442.15
			`mpfloat`	173.21	148.7	187.86	243.78	182.5
			filter + `real`	14.92	19.99	651.91	848.59	915.89
			filter + `gmp`	15.23	20.01	544.1	691.21	690.65
			filter + `mpfloat`	15.67	20.51	254.34	330.58	314.1
	10^6	h		*512*	*446817*	*999067*	*999775*	*999760*
		Sqrt	`double`	n/a				
			`real`	1600.34	3939.36	7147.13	7154.49	8563.14
			filter + `real`	201.16	297.92	4805.81	6774.37	8182.67
		Polyn.	`double`	55.46	94.93	n/a		
			`real`	1580.29	4402.52	8425.47	8658.88	8756.45
			`gmp`	4023.87	4974.26	6520.6	7141.51	5127.44
			`mpfloat`	1732.43	1868.43	2605.06	2999.01	2088.08
			filter + `real`	149.47	253.6	5765.34	8123.3	8805.28
			filter + `gmp`	153.88	255.03	4678.14	6636.59	6554.35
			filter + `mpfloat`	155.4	257.19	2375.97	3312.92	3260.81

Table 6
Experimental results for the OnLine data set. $T(n,h)$ is the total running time; n is the number of input sites and h the number of visible sites. b_{max} is the maximum bit size of the input; "n/a" implies that the algorithm could not compute a correct diagram due to numerical errors.

To give a rough idea of our code's speed, we note that the largest instance of these inputs can be solved exactly in about $5min$, $3h56min$, and $55min$, respectively (with filtered `mpfloat` arithmetic).

Let us now analyze Tables 4 (InSquare data), 5 (OnParabola) and 6

(OnLine). First, the small number of visible sites for small bit sizes for the OnParabola and OnLine inputs is due to the fact that we cannot represent more input sites than the bit size permits. The hidden inputs are just copies of the visible ones.

In the InSquare example the running times for all bit sizes $b > 10$ are approximately the same, irrespectively of the increase in the bit size of the input. This is a manifestation of the fact that interval arithmetic is largely sufficient for computing the Apollonius diagram, and that exact arithmetic is rarely used: interval arithmetic uses intervals of `double`s and thus the bit size of the input is irrelevant. The difference in running times between the case $b = 10$ and the cases $b > 10$ is due to the fact that in the former case we have many more hidden sites. More precisely, in the first case the radii have bit size $b' = 5 = b - 5$, whereas as for $b > 10$ the bit size of the radii is $b' = b - 10$. Therefore, the radii in the first case are larger than in the second, relatively, of course, to the coordinates of the centers.

Notice that the above observations do not apply to the OnParabola and Online inputs, for large bit sizes: the number of hidden sites is generally small compared to the number of visible sites. Moreover, these inputs are highly degenerate, thus exact arithmetic is much more often employed for the evaluation of the predicates, than for the InSquare inputs. As a result, the bit size of the input has a direct effect on the running times: as the bit size increases the running times for computing the Apollonius diagram also increases.

It is interesting that all data with small bit sizes can be treated exactly with double arithmetic, provided we apply the polynomial method. Let us restrict attention to inputs for which double arithmetic solves the problem correctly: the input bit size can be larger for the OnLine inputs than for the OnParabola inputs. This is due to the fact that the former inputs require predicates of maximum algebraic degree 6, whereas the latter of degree 14.

On the other hand, `double`, combined with the Sqrt method, was not able to compute a correct diagram for the OnLine example for any n and b_{max}. This means that the algorithm either cannot terminate, due to inconsistent evaluations, or yields an incorrect output. Another interesting situation is the case $b_{max} = 10$, $n = 10^6$, of the InSquare example, using again the Sqrt method. In this case we have a degeneracy w.r.t. OrderOnBisector, which is evaluated incorrectly and causes the algorithm to terminate prematurely. In short, comparing with the performance of `double` arithmetic offers an estimate of the price to be paid for achieving robustness.

A general observation concerns double arithmetic, compared to (filtered) exact number types. The fastest of the latter incurs an overhead factor of at most 4 in any of the examined data sets. A notable exception is the OnLine example with 10^4 sites, expressed by quantities of 30 bits. The slowdown factor is larger than 30. To explain this behavior, observe that it happens precisely at the largest input set which can be handled by `double` arithmetic. What becomes manifest here is the high cost of exception handling, as performed by the current implementation of filtered arithmetic. Exceptions are widely used to catch numerical errors, and significantly influence run time for large data sets, whereas for most inputs they constitute rare events. This is due to the fact that exceptions render a program highly non-sequential, in which case compiler optimizations cannot be effective enough. The remedy would be a more careful exception-free implementation, which requires an additional programming effort that we have not undertaken. Remark that, despite the fact that exceptions do not occur regularly, their handling becomes important for very large sizes.

A related observation concerns experiments with non-filtered `mpfloat` arithmetic: they confirm that this type is faster for OnParabola inputs of 50 bits than the filtered `mpfloat` type. For inputs of 40 bits the filtered and non-filtered `mpfloat` type take about the same time, and for inputs of 30 bits or less, the filtered `mpfloat` type is faster. This behavior for inputs of large bit-size should be attributed to the high cost of exception handling, which occurs every time the InCircle predicate is called, whereas for inputs of small bit-size, it is a combination of two facts: (i) interval arithmetic suffices to resolve the InCircle subpredicate, and (ii) the number of hidden sites is much higher relative to the inputs of larger bit-size, which implies that the InCircle subpredicate is called fewer times with respect to the number of input circles. Recall that determining if a site is hidden is done using expressions of algebraic degree up to 4. Similar observations can be made if we consider the filtered and non-filtered `real`, or the filtered and non-filtered `gmp` number types. The GMP package uses asymptotically faster algorithms than those supporting `mpfloat`. But this is not obvious from our experiments since they seldom treat sufficiently large instances. Finally, it is also interesting to see that we have an analogous pattern of behavior for the afore-mentioned filtered and non-filtered number types when we consider the OnLine inputs. The time gap, however, between the filtered and non-filtered number types is smaller, which should be attributed to the fact that the predicates evaluated for the OnLine inputs are of smaller degree than for the OnParabola inputs.

In comparing the most efficient implementation, with filtered number types, for each of the two predicate evaluation methods, we see that the Polynomial method is faster (with filtered `mpfloat` arithmetic) than the Sqrt method, on every instance. Their ratio of performance can be as large as 3, e.g., for the OnParabola inputs.

As before, we use two evaluation methods, denoted by the keywords Sqrt and Polynomial, respectively. The first one assumes that the operations $\{+, -, \times, /, \sqrt{\ }\}$ are performed exactly. The second uses the procedure of Figure 10 and assumes that only the operations $\{+, -, \times\}$ are performed exactly.

For reasons of programming simplicity concerning the most demanding primitive, namely RadiiDifference, we have used the approach sketched just after Theorem 13. In our current preliminary implementation, the factorization of R_0 to polynomials of degree 12 and 8 is not undertaken. The main reason is that this requires a large number of operations; e.g., the first factor is comprised of 205 monomials in the input parameters. Thus the maximum degree of the expressions actually tested by our program is 20.

We present two series of experiments. The first focuses on the entire algorithm. Two data sets are considered, one random and one in almost degenerate position (Table 7). The random set consists of $N = 5 \cdot 10^5$ sites with integer coordinates uniformly distributed in the square $[-M, M] \times [-M, M]$, where $M = 10^{14}$. The weights of the sites are integers uniformly distributed in the interval $[0, R]$, where $R = 10^{11}$. About 3% of the sites are hidden. The almost degenerate data set consists of N sites which are approximately tangent to the circle centered at the origin of radius M. The coordinates of the site centers are random integers. The radii of the sites are also random integers uniformly distributed in $[0, R]$. Less than 1% of the sites are hidden.

The second series of experiments focuses on the comparison between two algebraic numbers of degree 2 (Table 8). In particular, we are given the polynomials in (23) and we want to compare x_1^+ and x_2^+. We use 3 different methods, namely the one that represents the roots as radicals (cf. (24)), the method presented in [10] (cf. Fig. 9) and our method based on the evaluation tree of Figure 10. We use the keywords Sqrt, Dfmt and Polynomial, respectively. We consider 3 models for the bit size of the coefficients α_τ, β_τ and γ_τ. These are $(4b, 5b, 6b)$, $(b-2, b-1, b)$ and (b, b, b), where b is a parameter. The first one corresponds to our geometric problem, where b is the bit size of the inputs of our algorithm. The second and third model correspond to a homogeneous and a generic polynomial. The number types used are the same as in the first series of experiments. The experiments using `double`s are only given for reference. We consider polynomials with randomly chosen coefficients, using two scenarios: (1) the roots of the two polynomials are entirely independent (*random* data), and (2) the larger roots of the two polynomials are equal (*degenerate* data). No exception mechanism has been used; in other words, the code did not do any exception handling (no exception throwing and no exception catching).

Algorithm for Apollonius Diagram using hierarchy

	Method	Number type	Insertion	OrderOnBisector		RadiiDifference		x_1^+-x_2^+ tree	$P_3(\infty)$	P_4
			time	time	# calls	time	# calls	# calls	# evals	# evals
Random	Sqrt	`double`	60.63	1.62	50201	<0.01	10035	n/a	n/a	n/a
		`real`	761.17	19.8		0.04				
		filter + `real`	197.68	5.08		0.04				
	Polyn.	`double`	60.	1.5		0.02		8896	6198	6072
		`real`	778.59	20.32		0.12				
		`gmp`	3562.92	93.06		0.52				
		`mpfloat`	2493.55	66.54		0.97				
		filter + `real`	204.47	5.69		0.06				
		filter + `gmp`	190.15	5.35		0.08				
		filter + `mpfloat`	190.67	5.23		0.09				
Almost Degenerate	Sqrt	`real`	3291.61	129.6	2621874	14.97	1533280	n/a	n/a	n/a
		filter + `real`	3134.17	95.72	2622036	14.32	1533426			
	Polyn.	`real`	1712.26	93.64	2621874	12.31	1533280	767489	598460	825
		`gmp`	5282.56	402.89		62.04				
		`mpfloat`	3550.62	331.4		74.11				
		filter + `real`	1276.53	53.85	2622054	7.39	1533374	767523	598493	858
		filter + `gmp`	1178.41	53.1		8.18				
		filter + `mpfloat`	852.85	47.79		9.47				

Table 7
Comparison of different number types and methods for the overall algorithm and its predicates. Running times are in *seconds* and refer to the total time spent in the corresponding module; "# evals" denotes number of evaluations and "n/a" refers to cases where the corresponding entry is not applicable.

We observe that the running times of the filtered approach with random inputs are only 3 to 5 times larger than those with floating point arithmetic; this holds for all three tables. This may go against popular belief that assumes exact arithmetic to be excessively costly, when compared to numerical computation. The latter, moreover, offers no guarantee and would lead to inconsistencies when (near) degeneracies occur; in the case of Apollonius diagrams, inconsistencies may appear even with certain random inputs. Our experiments thus provide another confirmation that carefully implemented exact arithmetic imposes a reasonable overhead on efficiency.

An important observation is that the filtered approach is usually at least two times faster than the non-filtered approach, and more so for almost degenerate inputs; cf. Table 7. Among the filtered methods, our Sturm-based techniques run about twice as fast as the Sqrt method. It is natural that for almost degenerate inputs, the number of tests increases with filtering, because in certain cases an augmented precision is required. This increase occurs for both OrderOnBisector and RadiiDifference, and for both Sqrt and Polynomial methods, but not to the same extent. This is a manifestation

Methods for number types that support $\{+, -, \times, /, \sqrt{\ }\}$											
	Degree model $(\alpha_\mathcal{T}, \beta_\mathcal{T}, \gamma_\mathcal{T})$	b	`double`			`real`			filter + `real`		
			Sqrt	Dfmt	Polyn.	Sqrt	Dfmt	Polyn.	Sqrt	Dfmt	Polyn.
Random	$4b$, $5b$, $6b$	4	0.37	0.28	0.24	2.42	2.82	3.7	1.23	1.06	0.86
		8	0.24	0.26	0.32	3.63	5.35	5.1	1.39	0.84	0.79
	b-2, b-1, b	25	0.28	0.23	0.24	2.41	2.95	3.73	1.28	1.1	0.81
		50	0.35	0.31	0.22	4.17	5.5	5.24	1.33	0.91	0.79
	b, b, b	25	0.35	0.27	0.25	2.51	2.96	3.66	1.39	0.83	0.79
		50	0.21	0.25	0.26	3.66	5.69	5.49	1.33	0.99	0.98
Degenerate	$4b$, $5b$, $6b$	4	0.23	0.3	0.26	114.53	122.05	111.01	117.44	125.58	113.65
		8	0.29	0.23	0.34	485.95	421.69	367.17	488.23	426.66	372.79
	b-2, b-1, b	25	0.25	0.25	0.32	127.23	128.83	128.68	130.	132.58	131.23
		50	0.44	0.23	0.42	538.78	427.07	376.2	543.46	433.61	380.87
	b, b, b	25	0.27	0.24	0.33	126.84	129.41	129.62	130.6	132.51	131.68
		50	0.38	0.23	0.27	538.71	427.24	375.95	543.66	433.86	380.48

Methods for number types that support $\{+, -, \times\}$										
	Degree model $(\alpha_\mathcal{T}, \beta_\mathcal{T}, \gamma_\mathcal{T})$	b	`gmp`		`mpfloat`		filter + `gmp`		filter + `mpfloat`	
			Dfmt	Polyn.	Dfmt	Polyn.	Dfmt	Polyn.	Dfmt	Polyn.
Random	$4b$, $5b$, $6b$	4	25.95	23.25	11.65	9.69	0.97	0.81	0.89	0.81
		8	27.61	25.42	14.82	12.24	1.07	0.77	0.84	0.93
	b-2, b-1, b	25	25.63	23.7	12.83	9.68	0.88	0.87	0.84	0.75
		50	29.96	25.7	17.38	13.49	0.94	0.86	1.1	0.92
	b, b, b	25	26.2	23.04	12.84	10.07	1.08	0.9	0.95	0.95
		50	30.44	26.75	18.4	13.89	0.87	0.97	0.93	1.09
Degenerate	$4b$, $5b$, $6b$	4	43.16	37.68	19.42	15.78	53.33	46.26	28.53	23.58
		8	46.28	39.91	25.67	20.74	56.75	49.62	34.97	28.37
	b-2, b-1, b	25	44.01	37.9	20.54	15.59	53.8	46.88	28.7	24.37
		50	49.54	41.7	30.13	23.15	60.1	51.45	38.79	31.67
	b, b, b	25	43.82	37.8	20.53	17.03	53.24	46.71	29.83	24.57
		50	49.94	42.37	31.07	24.19	60.25	51.06	40.33	32.96

Table 8
Running times for the comparison of x_1^+ and x_2^+ considering various models and bit sizes for the coefficients. The measurements are in μsec and are the averages of 10^6 random input sequences.

of the fact that the two methods do not use the same error bounds; a larger input sample may shed light to this phenomenon.

It was expected that the number of tests should increase between random and almost degenerate inputs. It is less obvious, though, that this would not hold for the tests on P_4, the quantity examined at the maximum depth of the evaluation tree; cf. the last column of Table 7. Experimental data, not shown here, confirm that the output cases do not occur with the same frequency when the input is almost degenerate. Recall that the last 3 columns of Table

7 refer to those instances that require comparing x_1^+ and x_2^+, and different pairs of cases are distinguished by $P_3(\infty)$ and P_4. In our random data the prevailing cases are 3b ($J > 0$) and 4; hence the numerous P_4 tests. In the almost degenerate data the vast majority of cases is distributed between 2, 3b ($J < 0$) and 5. In this case $P_3(\infty)$ can decide the order of x_1^+, x_2^+ in about 99% of the cases for which K cannot yield an answer.

Another interesting corollary of our experiments is the clear improvement upon the method of [10], thanks to the procedure based on Sturm sequences, which reduces the number of operations; cf. Table 8. In addition, the inputs that do not require going to maximum depth of the evaluation tree have a lower algebraic degree, since we have replaced testing E by $P_3(\infty)$. This may have also allowed for better filtering.

Further conclusions can be drawn from Table 8 by considering the complexity as a function of bit size. We observe that, for random inputs, the cost increases in sublinear fashion, because several tests can be performed accurately at low precision. As far as degenerate inputs are concerned, this is again true with multi-precision integer or floating-point arithmetic, which is adaptive in this sense. However, using reals implies a superlinear dependence on bit size because of lazy evaluations, which require that several evaluations be repeated as precision increases. The GMP package uses asymptotically faster algorithms than those supporting `MP_Float`. But this is not obvious from our experiments since they never treat quantities of more than about 200 bits.

11 Extensions and future work

As we have mentioned in the introduction, our algorithmic analysis of the predicates for the Apollonius diagram can be almost immediately adapted for the algorithm in [4]. The latter algorithm needs a single predicate: given an edge of the Apollonius diagram, defined by four sites B_i, B_j, B_k and B_ℓ, as well as a fifth site B_m, determine what portion of the edge $\alpha_{ij}^{k\ell}$ is destroyed by the new site B_m. This is, essentially, our EdgeConflictType predicate. In our case, however, when the EdgeConflictType predicate is called we have already verified that B_m is not a hidden site. Hence, in order to use the EdgeConflictType predicate we need to slightly modify it as follows. We first check if B_m is contained in the interior of any of the four sites B_ν, $\nu = i, j, k, \ell$ (clearly, if one of these sites is the site at infinity B_∞, B_m is not contained in the interior of that site). If this is the case, then B_m is hidden and thus B_m is not in conflict with the edge $\alpha_{ij}^{k\ell}$. Otherwise, and depending on whether the edge $\alpha_{ij}^{k\ell}$ lies on an infinite or finite bisector, we simply call the FiniteEdgeConflictType or InfiniteEdgeConflictType predicate. Clearly, the algebraic degree of the predicate required by the algorithm [4] is 16.

It is also interesting to notice that our algorithmic analysis of the predicates for the Apollonius diagram is also applicable to the case of smooth disjoint convex objects. More specifically, consider the scenario, where instead of circular sites we have smooth convex objects (e.g., ellipses) that are disjoint. The distance of a point on the plane from an object is now defined to be the minimal distance from the point to the boundary of the convex object. The distance is positive if the point lies in the complement of the convex object, zero if it lies on its boundary, and negative if it lies in its interior. It turns out that under this distance metric, the Voronoi diagram of the convex objects is a special case of an abstract Voronoi diagram [31], and we can either use the dynamic algorithm presented in [31] or the one for abstract Voronoi diagrams in [4]. The second algorithm would simply require the EdgeConflictType predicate (since the objects are disjoint we do not have hidden sites). The predicates required for the first algorithm, when applied to disjoint convex objects are the OrderOnBisector, EdgeConflictType, Orientation and IsDegenerateEdge predicates. Their decomposition to subpredicates and primitives described in Section 3.3 is still valid. Naturally, the way the Orientation predicate, the DistanceFromBitangent and InCircle subpredicates, and the χ_2, Existence and RadiiDifference primitives are computed will now depend on the type of convex objects considered.

Given our remarks above we would like to apply our algorithmic analysis of the predicates to the computation of the 2D Voronoi diagram of specific instances of smooth convex objects, such as ellipses or C^1-continuous composite Bézier curves. We also expect that it will be possible to use Sturm sequences for the exact computation of arrangements of curves, as well as to extend our approaches to the three dimensional problem, i.e., the Apollonius diagram of spheres in 3D. Some preliminary efforts in the direction of computing exactly single cells of the 3D Apollonius diagram have been made in [32] and [33,34]. In [32] the implementation is not exact, but rather relies on controlled floating point arithmetic; when the floating-point arithmetic fails the input is numerically perturbed and the computation is restarted. The implementation in [33,34] is exact and relies on the computation of a special two-dimensional Voronoi diagram, called Möbius diagram, that is combinatorially equivalent to a single 3D Apollonius cell (cf. [35]).

Acknowledgments

We thank Ron Wein and Nicola Wolpert for a long list of helpful comments on the presentation as well as for catching certain typos. We also thank Monique Teillaud and Sylvain Pion for helpful discussions and constructive comments on preliminary versions of this paper. Finally, we thank Olivier Devillers for letting us use the implementation of the methods in [10].

The first author was partially supported by INRIA's "Calamata" project, a bilateral collaboration between the National Kapodistrian University of Athens and the Galaad group at INRIA Sophia-Antipolis, and by "Pythagoras" (number 70/3/7392), a project under the EPEAEK program of the Greek Ministry of Educational and Religious Affairs. Both authors were partially supported by the IST Programme of the EU as a Shared-cost RTD (FET Open) Project IST-2000-26473 (ECG - Effective Computational Geometry for Curves and Surfaces).

References

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

[2] R. Drysdale, III, D. Lee, Generalized Voronoi diagrams in the plane, in: Proc. 16th Allerton Conf. Commun. Control Comput., 1978, pp. 833–842.

[3] S. Fortune, A sweepline algorithm for Voronoi diagrams, in: Proc. 2nd Annu. ACM Sympos. Comput. Geom., 1986, pp. 313–322.

[4] R. Klein, K. Mehlhorn, S. Meiser, Randomized incremental construction of abstract Voronoi diagrams, Comput. Geom. Theory Appl. 3 (3) (1993) 157–184.

[5] M. Sharir, Intersection and closest-pair problems for a set of planar discs, SIAM J. Comput. 14 (1985) 448–468.

[6] F. Anton, J.-D. Boissonnat, D. Mioc, M. Yvinec, An exact predicate for the optimal construction of the additively weighted Voronoi diagram, in: Proc. 18th Europ. Workshop Comput. Geom., 2002.

[7] D.-S. Kim, D. Kim, K. Sugihara, Voronoi diagram of a circle set from Voronoi diagram of a point set: I. Topology, CAGD 18 (2001) 541–562.

[8] D.-S. Kim, D. Kim, K. Sugihara, Voronoi diagram of a circle set from Voronoi diagram of a point set: II.Geometry, CAGD 18 (2001) 563–585.

[9] M. Karavelas, M. Yvinec, Dynamic additively weighted Voronoi diagrams in 2D, in: Proc. 10th European Symp. on Algorithms (ESA), Vol. 2461 of LNCS, 2002, pp. 586–598.

[10] O. Devillers, A. Fronville, B. Mourrain, M. Teillaud, Algebraic methods and arithmetic filtering for exact predicates on circle arcs, Comp. Geom: Theory & Appl., Spec. Issue 22 (2002) 119–142.

[11] N. Geismann, M. Hemmer, E. Schömer, Computing a 3-dimensional cell in an arrangement of quadrics: Exactly and actually!, in: Proc. Annual ACM Symp. on Comput. Geometry, 2001, pp. 264–273.

[12] E. Berberich, A. Eigenwillig, M. Hemmer, S. Hert, K. Mehlhorn, E. Schömer, A computational basis for conic arcs and boolean operations on conic polygons, in: Proc. 10th European Symp. on Algorithms (ESA), Vol. 2461 of LNCS, 2002, pp. 174–186.

[13] R. Wein, High-level filtering for arrangements of conic arcs, in: Proc. 10th European Symp. on Algorithms (ESA), Vol. 2461 of LNCS, 2002, pp. 884–895.

[14] N. Wolpert, Jacobi curves: Computing the exact topology of arrangements of non-singular algebraic curves, in: Proc. 11th European Symp. on Algorithms (ESA), Vol. 2832 of LNCS, 2003, pp. 532–543.

[15] M. Karavelas, I. Emiris, Root comparison techniques applied to the planar additively weighted Voronoi diagram, in: Proc. 14th ACM-SIAM Symp. on Discrete Algorithms (SODA), 2003, pp. 320–329.

[16] D. Perrucci, Different methods to compare real roots of polynomials of degree 2 and their extensions to degree 4, Technical Report ECG-TR-242200-03, INRIA Sophia-Antipolis (2003).

[17] I. Emiris, E. Tsigaridas, Computing with real algebraic numbers of small degree, in: Proc. European Symp. on Algorithms, LNCS, Springer Verlag, 2004, pp. 652–663.

[18] M. Karavelas, Proximity structures for moving objects in constrained and unconstrained environments, Ph.D. thesis, Stanford University (2001).

[19] P. Angelier, Algorithmique des graphes de visibilité, Ph.D. thesis, Université Paris VII (2002).

[20] D. Cox, J. Little, D. O'Shea, Using Algebraic Geometry, no. 185 in Graduate Texts in Mathematics, Springer-Verlag, New York, 1998.

[21] I. Emiris, B. Mourrain, Matrices in elimination theory, J. Symbolic Computation, Special Issue on Elimination 28 (1999) 3–44.

[22] B. Sturmfels, Algorithms in Invariant Theory, RISC Series on Symbolic Computation, Springer Verlag, Vienna, 1993.

[23] B. Buchberger, G. Collins, R. Loos (Eds.), Computer Algebra: Symbolic and Algebraic Computation, 2nd Edition, Vol. 4 of Computing Supplementum, Springer-Verlag, Wien, 1982.

[24] C. Yap, Fundamental Problems of Algorithmic Algebra, Oxford University Press, New York, 2000.

[25] R. Churchill, Complex variables and applications, 2nd Edition, McGraw-Hill Book Co., 1960.

[26] H. Edelsbrunner, Algorithms in Combinatorial Geometry, Springer-Verlag, Heidelberg, 1987.

[27] R. Seidel, The nature and meaning of perturbations in geometric computing, in: Proc. 11th Symp. Theoret. Aspects Comp. Sc., LNCS 775, Springer-Verlag, 1994, pp. 3–17.

[28] T. Granlund, GMP, the GNU multiple precision arithmetic library.
URL `http://www.swox.com/gmp/`

[29] S. Pion, Interval arithmetic: An efficient implementation and an application to computational geometry, in: Workshop on Appl. of Interval Analysis to Systems and Control, 1999, pp. 99–110.
URL `www-sop.inria.fr/prisme/biblio/search.html`

[30] V. Karamcheti, C. Li, I. Pechtchanski, C. Yap, The CORE Library Project, 1st Edition (1999).
URL `http://www.cs.nyu.edu/exact/core/`

[31] M. Karavelas, M. Yvinec, The Voronoi diagram of convex objects in the plane, in: Proc. 11th European Symp. on Algorithms (ESA), Vol. 2832 of LNCS, 2003, pp. 337–348.

[32] H.-M. Will, Fast and efficient computation of additively weighted voronoi cells for applications in molecular biology, in: Proc.6th Skandinavian Workshop on Algorithm Theory (SWAT), Vol. 1432 of LNCS, 1998, pp. 310–321.

[33] C. Delage, CGAL-based first prototype implementation of Möbius diagram in 2D, Technical Report ECG-TR-241208-01, INRIA Sophia-Antipolis (2003).

[34] C. Delage, Diagrammes de Möbius en dimension 2, Master's thesis, INRIA Sophia-Antipolis (2003).

[35] J.-D. Boissonnat, M. I. Karavelas, On the combinatorial complexity of Euclidean Voronoi cells and convex hulls of d-dimensional spheres, in: Proc. 14th ACM-SIAM Symp. on Discrete Algorithms (SODA), 2003, pp. 305–312.