Smallest Enclosing Disk Queries Using Farthest-Point Voronoi Diagrams

Kevin Buchin
Department of Computer Science, TU Dortmund University, Germany

Mark Joachim Krallmann
Department of Computer Science, TU Dortmund University, Germany

Frank Staals
Department of Information and Computing Sciences, Utrecht University, The Netherlands

Abstract

Let S be a set of n points in $\mathbb{R}^2$. Our goal is to preprocess S to efficiently compute the smallest enclosing disk of the points in S that lie inside an axis-aligned query rectangle. Previous data structures for this problem achieve a query time of $O(\log^6 n)$ with $O(n \log^2 n)$ preprocessing time and space by lifting the points to 3D, dualizing them into polyhedra, and searching through their intersections. We present a significantly simpler approach, solely based on 2D geometric structures, specifically 2D farthest-point Voronoi diagrams. Our approach achieves a deterministic query time of $O(\log^4 n)$ and, via randomization, an expected query time of $O(\log^{5/2} n \log\log n)$ with the same preprocessing bounds.

2012 ACM Subject Classification Theory of computation → Computational geometry; Theory of computation → Design and analysis of algorithms; Theory of computation → Data structures design and analysis

Keywords and phrases Range searching, smallest enclosing disk, farthest point Voronoi diagram

Digital Object Identifier 10.4230/LIPIcs.ESA.2026.124

Related Version *Full Version*: `https://arxiv.org/abs/2605.00743` [9]

Funding *Mark Joachim Krallmann*: Funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) - 550797858.

1 Introduction

Let S be a set of n points in $\mathbb{R}^2$. Computing the smallest radius disk that contains all points in S, i.e. the *smallest enclosing disk* $\mathrm{SED}(S)$ is a fundamental and well-studied problem in computational geometry that has various applications in facility location and clustering [3, 7, 14]. Megiddo gave a linear-time algorithm to compute the smallest enclosing disk (or even the smallest enclosing ball of points in $\mathbb{R}^d$) [21]. Furthermore, Welzl's simple, randomized incremental construction algorithm [26] now appears in various text books on computational geometry [11]. However, in many applications we are not interested in the smallest enclosing disk of *all* points in S, but only of the points in an area of interest. For example, a data analyst may want to interactively select some region Q, and quickly compute the smallest enclosing disk of the points in $S \cap Q$. See Figure 1. Hence, we want to build a data structure that can answer such *smallest enclosing disk queries* efficiently.

Our smallest enclosing disk queries are a form of *range aggregation queries*: i.e. store the set S so that one can efficiently compute some aggregation function $f(S \cap Q)$ of the points in a query range Q. Simple aggregation functions such as counting or reporting the number of points in the query range are extremely well studied [10, 11, 22]. More recently, there is an interest in more advanced aggregation functions. Examples include queries that ask for (approximate) heavy hitters and quantiles [2], color frequency reporting [16],

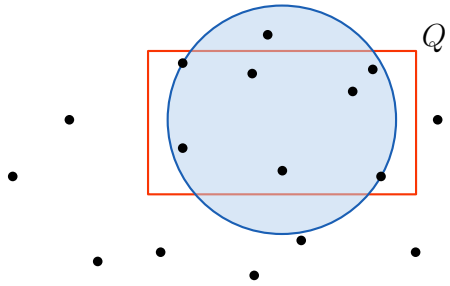


Figure 1 A rectangular query region Q and the smallest enclosing disk of $S \cap Q$.

(approximate) smallest enclosing disks [8, 20] (reviewed in more detail below) and balls [18], or even queries that ask for an entire (approximate) k-center, k-median, or k-means clustering of $S \cap Q$ [1, 24]. Many of these approximate results rely on composable coresets [17]; i.e. the ability to compactly summarize a subset of the points, so that we can combine the summaries to obtain a summary of $S \cap Q$ [23].

Smallest enclosing disk queries. We will focus on smallest enclosing disk queries in the case that the query ranges are axis-aligned rectangles. There are $\Theta(n^4)$ combinatorially distinct query ranges, so clearly we could precompute and store all answers (taking $O(n^5)$ time $O(n^4)$ space) so that we can answer queries in optimal $O(\log n)$ time. However, this is prohibitively large. Alternatively, in (near) linear time and space one can build a data structure to answer range reporting queries (e.g. a range tree or kd-tree) to report the k points in $S \cap Q$ and then compute $\text{SED}(S \cap Q)$ in additional $O(k)$ time. However, as the number of points k in the query range may be large, this query time (e.g. $O(\log n + k)$) may not be sufficient for interactive applications.

Unfortunately, computing the smallest enclosing disk is not decomposable; i.e. one cannot easily compute $\text{SED}(A \cup B)$ from $\text{SED}(A)$ and $\text{SED}(B)$. Hence, one cannot directly use similar ideas as for answering e.g. range counting queries. Instead, in [8] the authors argue that point sets A and B can be lifted and dualized into convex polyhedra P_A and P_B, so that one can compute $\text{SED}(A \cup B)$ by searching in (an implicit representation of) the intersection of P_A and P_B. Such a searching query is somewhat similar to answering linear programming query, and thus, given m convex polyhedra of total complexity n, one can answer a query in $O(m^3 \log^3 n)$ time [15]. By combining this machinery with a standard 2D-range tree, one can obtain $S \cap Q$ as $m = O(\log^2 n)$ appropriately preprocessed canonical subsets, and thus answer a smallest enclosing disk query in $O(\log^9 n)$ time. The data structure uses $O(n \log^2 n)$ space and can be built in $O(n \log^2 n)$ time [8]. In [20] the authors presented a modified range tree that allows to reduce the number of considered canonical sets to $O(\log n)$, thereby improving the query time to $O(\log^6 n)$.

Our Results. We will show that we can instead decompose the problem by using farthest-point Voronoi diagrams. The farthest point Voronoi diagram $\text{FPVD}(S)$ of S is a subdivision of the plane into interiorly disjoint maximal regions so that any point q in a region $\text{FPVC}(p, S)$ has the same site p that is farthest from q among S. We show that given the farthest-point Voronoi diagrams of A and B (in an appropriately preprocessed form), we can find the smallest enclosing disk $\text{SED}(A \cup B)$ in $O(\log^2 n)$ time (where $n = |A| + |B|$). Our procedure, described in Section 3 extends to m diagrams; and runs in $O(m^2 \log^2 n)$ time. This allows us to bypass the whole lifting and dualizing to step, as well as the somewhat intricate searching among

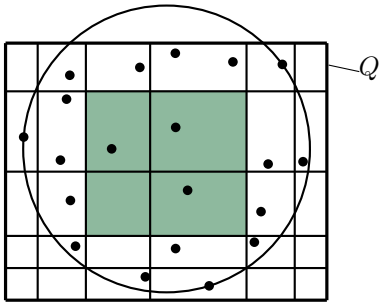


Figure 2 Based on [20]. To compute $\mathrm{SED}(S \cap Q)$ we can discard the green-shaded nodes.

the intersection of 3D convex polyhedra steps used by the earlier approaches [8, 20]. Hence, this significantly simplifies the query procedure. Furthermore, it improves the query time to $O(\log^4 n)$. The preprocessing step also remains relatively simple. One may furthermore wonder whether one can use randomization to further simplify the algorithm (as was the case in the "base" problem). Unfortunately, we show that one cannot straightforwardly generalize Welzl's randomized incremental construction algorithm for computing the smallest enclosing disk of points to computing the smallest enclosing disk of disjoint convex polygons (corresponding to our canonical subsets). See Section 4. However, we can combine our earliest farthest-point Voronoi diagram based machinery with a randomized dynamic programming based framework from Eppstein [15] to further speed up smallest enclosing disk queries. In particular, our data structure answers such queries in expected $O(\log^{5/2} n \log\log n)$ time. The preprocessing time and space remain $O(n \log^2 n)$.

2 Preliminaries

In this section, we briefly review some of the basic tools and techniques that we build upon throughout the paper. Let S be a set of n points in $\mathbb{R}^2$. We assume that the points in S are in general position, i.e. no three points are collinear and no four points are cocircular. If this assumption is violated, we use symbolic perturbation [13]. Let $\mathrm{CH}(S) \subseteq S$ denote the vertices of the convex hull of S and $\mathrm{mid}(p, q)$ denote the midpoint of two points $p, q \in S$.

2.1 Range-aggregate queries for smallest enclosing disk

Our goal is to preprocess and store S, such that we can efficiently compute the smallest enclosing disk $\mathrm{SED}(S \cap Q)$ of an axis aligned query rectangle $Q = [x, x'] \times [y, y'] \subseteq \mathbb{R}^2$. Our goal is to answer such queries in $O(\mathrm{polylog}\, n)$ time using $O(n \,\mathrm{polylog}\, n)$ space.

We achieve this by building the modified 2-dimensional range tree of [20] on S. Compared to a standard range tree, this structure yields only $O(\log n)$ canonical nodes $v_1, \ldots, v_m$ to consider, instead of $O(\log^2 n)$. The key observation, as described in [20], is that $\mathrm{SED}(S \cap Q)$ is determined solely by vertices of the convex hull. All nodes that cannot contribute any point to the convex hull can be discarded, as shown in Figure 2. We represent $S \cap Q$ with the corresponding canonical sets $S_1 = P(v_1), \ldots, S_m = P(v_m)$ such that $\mathrm{CH}(S_1 \cup \cdots \cup S_m) = \mathrm{CH}(S \cap Q)$.

We can partition $\mathrm{CH}(S \cap Q)$ into maximal sections contributed by one canonical set respectively. Let $p_1, \ldots, p_k$ be the points in $\mathrm{CH}(S)$ in clockwise order (beginning with the point that is smallest lexicographically), then we refer to the set of points $\{p_a, p_{a+1}, \ldots, p_{b-1}, p_b\}$, wrapping around if needed, with $\mathrm{CH}(S)_{a:b}$. We refer to $\mathrm{CH}(S)_{a:b}$ as a *section* of $\mathrm{CH}(S)$.

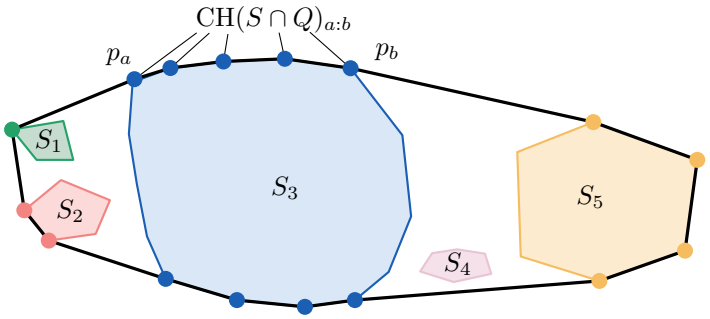


Figure 3 The convex hull consists of canonical sections, such as $\mathrm{CH}(S_1)_{a:b}$.

We may include or exclude points relative to p_a, e.g. by writing $\mathrm{CH}(S)_{a+1:b-1}$ we exclude p_a and p_b. Note that a section such as $\mathrm{CH}(S)_{a+1:b-1}$ may be empty if there are no points between p_a and p_b in the clockwise sequence. Let $S_1, \ldots, S_m$ be canonical sets. We refer to a non-empty section $\mathrm{CH}(S_i)_{a:b}$ as a *canonical section* of S_i when there is an equivalent section $\mathrm{CH}(S \cap Q)_{a':b'} = \mathrm{CH}(S_i)_{a:b}$ that is maximal, i.e. S_i does not include the clockwise and counter-clockwise neighbors, hence $\mathrm{CH}(S \cap Q)_{a'-1:b'} \not\subseteq S_i$ and $\mathrm{CH}(S \cap Q)_{a':b'+1} \not\subseteq S_i$.

Figure 3 illustrates an example, where $\mathrm{CH}(S \cap Q)_{a:b} \subseteq \mathrm{CH}(S_3)$. Observe that the set S_3 has two canonical sections, while the other sets have one, or none in case of S_4. We can bound the total number of canonical sections.

▶ **Lemma 1.** *The convex hull of $S \cap Q$ consists of $O(m)$ canonical sections.*

The proof for this lemma (and other omitted proofs) can be found in the full version [9]. To derive canonical sections we compute tangents connecting sections of $\mathrm{CH}(S \cap Q)$ in time $O(m^2 \log n)$, i.e. time $O(\log^3 n)$ with $m = O(\log n)$ as shown in [8].

During preprocessing, for every secondary range tree we compute the convex hulls of canonical sets bottom up, by merging the convex hulls of children in linear time. Thus, we require $O(n \log^2 n)$ preprocessing time storage for the complete tree.

2.2 Farthest-point Voronoi diagrams

A well-known data structure that has close connections to the smallest enclosing disk is the *farthest-point Voronoi diagram* [11]. The farthest-point Voronoi diagram assigns the farthest point of S to every point in the plane, i.e. the point $p \in S$ such that $d(q,p) = \max_{p' \in S} d(q,p')$, as illustrated by Figure 4. With $\mathrm{FPVD}(S)$ we refer to the farthest-point Voronoi diagram of the set of points S. The diagram $\mathrm{FPVD}(S)$ is a subdivision of the plane into $O(|S|)$ cells, such that the farthest point does not change in a cell. Every point $p \in \mathrm{CH}(S)$ has exactly one non-empty cell $\mathrm{FPVC}(p,S) = \bigcap_{q \in S \setminus \{p\}} H(p,q)$ where $H(p,q) = \{x \in \mathbb{R}^2 \mid d(x,q) \leq d(x,p)\}$ is the bisector-induced closed halfspace in which p is at least as far. Points in $S \setminus \mathrm{CH}(S)$ have no cell. In the remainder of the paper, we consider only points $p \in \mathrm{CH}(S)$ that have a cell.

A farthest-point Voronoi diagram can be interpreted as a tree of linear complexity [11] as illustrated by Figure 4. A vertex of $\mathrm{FPVD}(S)$ is a point $v \in \mathbb{R}^2$, such that there are three points $p_1, p_2, p_3 \in S$ with $\{v\} = \mathrm{FPVC}(p_1,S) \cap \mathrm{FPVC}(p_2,S) \cap \mathrm{FPVC}(p_3,S)$. We refer to p_1, p_2, p_3 as the *defining* points of v. An edge e of $\mathrm{FPVD}(S)$ is a (half-infinite) line segment, such that two points $p_1, p_2 \in S$ exist with $e = \mathrm{FPVC}(p_1,S) \cap \mathrm{FPVC}(p_2,S)$ and the intersection is non-degenerate (i.e. contains more than one point). We call p_1 and p_2 the *defining* points of e. Let $e_1, \ldots, e_m$ be the edges that bound $\mathrm{FPVC}(p,S)$ in clockwise order. Let p_i be the *defining* point of e_i; i.e. the point that together with p defines e_i. We then use the following key observations:

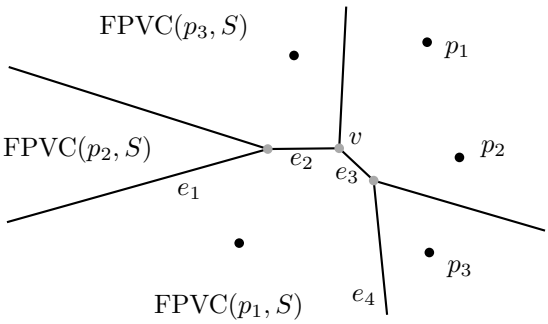


Figure 4 A farthest-point Voronoi diagram consisting of vertices, such as v and edges such as $e_1, \ldots, e_4$. The edges e_1 and e_4 are half-infinite edges.

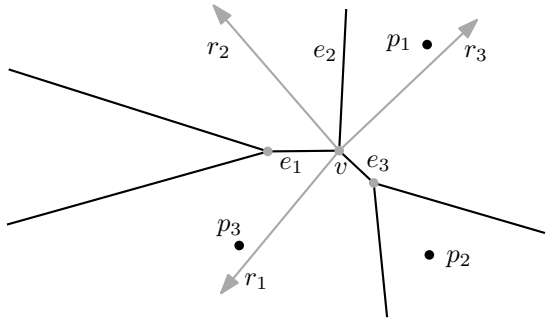


Figure 5 The defining points of a vertex admit rays that divide the plane and FPVD(S) in three parts.

▶ **Observation 2.** *Let r be a ray starting at any point $s \in \mathrm{FPVC}(p, S)$ and pointing opposite to $p \in S$. The ray r is fully contained in $\mathrm{FPVC}(p, S)$; i.e. $r \subseteq \mathrm{FPVC}(p, S)$.*

Observation 2 also implies that each cell is unbounded and hence also proves that the graph FPVD(S) does not contain circles [11, 6]. It then follows that e_1 and e_m are half-infinite edges (halflines), and all other edges e_i are bounded line segments. Furthermore:

▶ **Observation 3.** *The defining point p_1 for the first edge e_1 of any cell $\mathrm{FPVC}(p, S)$ is the clockwise neighbor of p on $\mathrm{CH}(S)$. Similarly, the defining point p_m for the last edge e_m is the counter-clockwise neighbor of p on $\mathrm{CH}(S)$.*

▶ **Observation 4.** *Let $e_1, \ldots, e_m$ be the edges of $\mathrm{FPVC}(p, S)$ in clockwise order, then the corresponding defining points $p_1, \ldots, p_m$ are also in clockwise order on the convex hull.*

▶ **Observation 5.** *Let p_1, p_2, p_3 be three defining points of a vertex v, and let r_i be the ray starting at v pointing opposite to p_i. The three rays r_1, r_2, r_3 divide the plane and $\mathrm{FPVD}(S)$ in three disjoint parts. See Figure 5.*

The last observation regards the subgraph induced by a section $\mathrm{CH}(S_i)_{a:b}$. Let G be the subgraph consisting of all edges and vertices that are incident to a cell of a point of $\mathrm{CH}(S_i)_{a:b}$. This subgraph has just one component, i.e. is a subtree: The cells of neighboring points of $\mathrm{CH}(S_i)_{a:b}$ share a half-infinite edge, hence lie in the same component of G. Since G contains the boundary of every cell, all cells are connected transitively.

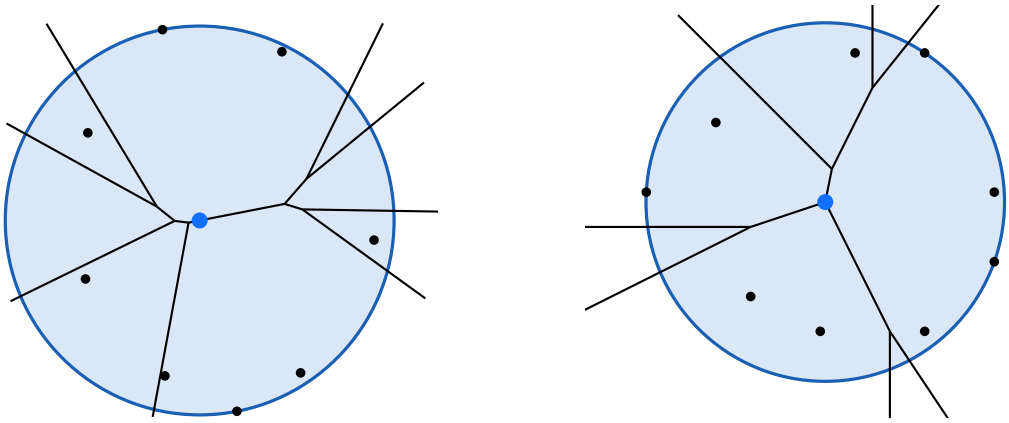


Figure 6 The center (blue) of SED(S) lies on an edge of FPVD(S) (left) or is a vertex (right).

▶ **Observation 6.** *For any set of points S and section of the convex hull $\mathrm{CH}(S)_{a:b}$, the edges and vertices of $\mathrm{FPVD}(S)$ that are adjacent to cells of points of $\mathrm{CH}(S)_{a:b}$ form a subtree.*

Farthest-point Voronoi diagrams and smallest enclosing disks. A well known fact regards the points defining $\mathrm{SED}(S)$.

▶ **Fact 7.** *The smallest enclosing disk is defined by two or three points lying on its boundary, forming either an antipodal pair or a non-obtuse triangle.*

Since the defining points lie on the boundary, they are the farthest points from the center z of $\mathrm{SED}(S)$. If there are three defining points p_a, p_b and p_c, then z is a vertex of $\mathrm{FPVD}(S)$ defined by p_a, p_b and p_c. In case of an antipodal pair p_a and p_b, the center z lies on the edge defined by p_a and p_b and corresponds to their midpoint $\mathrm{mid}(p_a, p_b)$. Figure 6 presents two examples. Hence, the farthest-point Voronoi diagram can be interpreted as a candidate set of linear size. In fact the first subquadratic algorithm [25] for smallest enclosing disk utilizes this property to determine $\mathrm{SED}(S)$ in time $O(n \log n)$.

For every node v of a secondary range tree we compute $\mathrm{FPVD}(P(v))$ in time $O(|P(v)|)$ based on the convex hull [4]. The bound of $O(n \log^2 n)$ preprocessing time and storage for the complete tree follows with a straightforward analysis. Based on [20, 8] we then have:

▶ **Lemma 8.** *Using $O(n \log^2 n)$ time and space we can build a data structure that given a query Q in time $O(\log^2 n)$ yields $m = O(\log n)$ canonical sets representing $\mathrm{CH}(S \cap Q)$ including their convex hulls and farthest-point Voronoi diagrams. In time $O(m^2 \log n)$ we can derive $O(m)$ canonical sections of $\mathrm{CH}(S \cap Q)$.*

3 Answering a smallest enclosing disk query

In this section, we show how we can efficiently answer smallest enclosing disk queries using farthest-point Voronoi diagrams. In Subsection 3.1 we show that for any edge of $\mathrm{FPVD}(S)$ we can decide which subtree contains the center of the smallest enclosing disk. This leads to a $O(\log n)$ procedure to compute $\mathrm{SED}(S)$. In the context of range-aggregate queries this search procedure is not immediately applicable, since we do not have access to the diagram $\mathrm{FPVD}(S \cap Q)$. Instead, we can get the points on $\mathrm{CH}(S \cap Q)$ as $O(\log n)$ canonical sections each of which defines a subtree of $\mathrm{FPVD}(S \cap Q)$. In Subsection 3.2 we develop a search procedure to identify the defining points contained in a given canonical section $\mathrm{CH}(S_i)_{a:b}$.

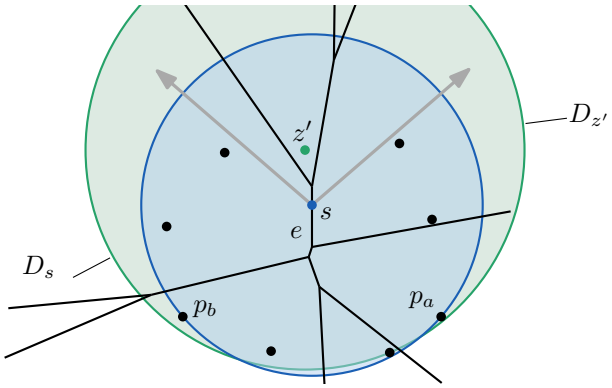


Figure 7 Any enclosing disk centered above the gray rays, such as $D_{z'}$ is larger than D_s.

This procedure requires accessing edges of $\mathrm{FPVD}(S \cap Q)$. We describe how we can implement these operations efficiently in Subsection 3.3. This then results in a $O(\log^4 n)$ time algorithm for querying $\mathrm{SED}(S \cap Q)$, as we argue in Subsection 3.4.

3.1 Determining the smallest enclosing disk for one set

Let S be a set of points. An edge or vertex of $\mathrm{FPVD}(S)$ contains the center z of $\mathrm{SED}(S)$. Since $\mathrm{FPVD}(S)$ is a tree, from any point on an edge or vertex of $\mathrm{FPVD}(S)$ there is unique path to z. Given an edge e of $\mathrm{FPVD}(S)$, we can decide which of the two subtrees connected by e contains z using the following lemma.

▶ **Lemma 9.** *Let e be an edge of $\mathrm{FPVD}(S)$ and p_a, p_b be the defining points of e, then for any $s \in e$ with $s \neq \mathrm{mid}(p_a, p_b)$ the subtree of $\mathrm{FPVD}(S)$ reachable over e by starting on s and moving away from $\mathrm{mid}(p_a, p_b)$ does not contain the center of $\mathrm{SED}(S)$.*

Proof. Let z be the center of $\mathrm{SED}(S)$. Consider the rays r_a and r_b emanating from s in the directions opposite to p_a and p_b, respectively as shown in Figure 7. The rays induce two regions in the plane. Let R be the region not containing $\mathrm{mid}(p_a, p_b)$. Note that any point $z' \in R$ is at least as far to p_a or p_b as s, i.e. $\max\{d(z', p_a), d(z', p_b)\} \geq d(s, p_a) = d(s, p_b)$. Hence, the enclosing disk $D_{z'}$ centered on z' is at least as large as the enclosing disk D_s, centered at s. In case $s \neq z$, i.e. D_s is not optimal, we also have $z' \neq z$ by transitivity and conclude $z \notin R$. In case $s = z$, observe that p_a, p_b are not antipodal since $s \neq \mathrm{mid}(p_a, p_b)$ by assumption. Thus s corresponds to an incident vertex of e, in particular to the vertex closer to $\mathrm{mid}(p_a, p_b)$, since the other vertex admits the larger disk. In either case the subtree reachable by moving away from $\mathrm{mid}(p_a, p_b)$ does not contain z and thus the statement follows. ◀

To *analyze an edge*, i.e. to decide which subtree contains the center for an edge, it suffices to know the defining points and a single point on the edge.

▶ **Lemma 10.** *Given (s, p_a, p_b), where $p_a, p_b \in S$ define an edge e of $\mathrm{FPVD}(S)$ such that $s \in e$ we can decide in time $O(1)$ that s is the center of $\mathrm{SED}(S)$ or which of the sections $\mathrm{CH}(S)_{a+1:b-1}, \mathrm{CH}(S)_{b+1:a-1}$ contains no defining points of $\mathrm{SED}(S)$.*

Proof. We check $s = \mathrm{mid}(p_a, p_b)$, if this holds then s is the center of $\mathrm{SED}(S)$. By Observation 6 the sections $\mathrm{CH}(S)_{a+1:b-1}, \mathrm{CH}(S)_{b+1:a-1}$ induce subtrees of $\mathrm{FPVD}(S)$. They are connected by e. By comparing the relative positions of s and $\mathrm{mid}(p_a, p_b)$ by Lemma 9 we infer which section does not contain defining points of $\mathrm{SED}(S)$. ◀

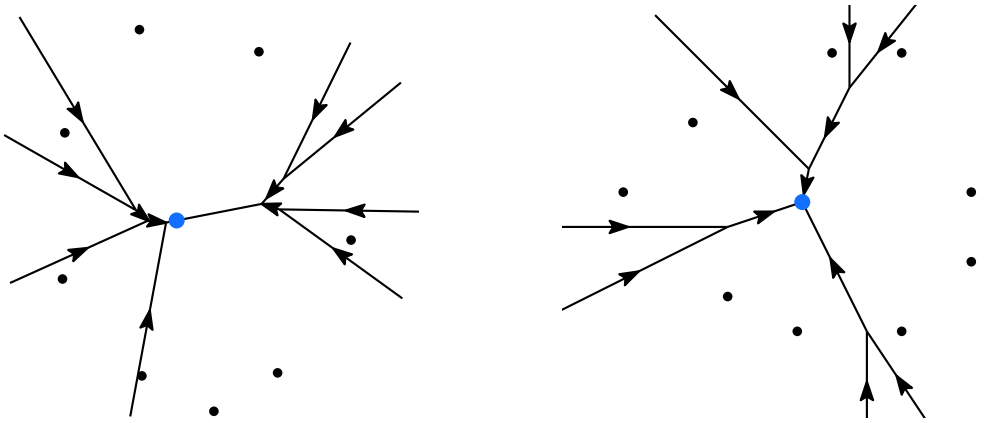


Figure 8 Farthest-point Voronoi diagrams, where SED(S) is defined by two points (left) or three points (right). The edges are annotated according to Lemma 9.

This allows us to determine SED(S) based on simple comparisons of midpoints to edges. Figure 8 illustrates how every edge guides towards the center of SED(S). We could develop a procedure that starts at an arbitrary vertex of FPVD(S) and traverses the diagram until it encounters the edge or vertex defined by the defining points of SED(S) in time $O(n)$. However, we aim for a more efficient procedure.

To achieve this, we require a search data structure that allows us to discard entire subtrees of FPVD(S) efficiently. One suitable data structure is the centroid decomposition, which we introduce next.

Let T be a tree of size n, then a *centroid* of T is a vertex v of T, such that removing v splits T into subtrees having a maximum size of $\frac{n}{2}$ each. The existence of centroids was shown in 1869 by Jordan [19]. We obtain the *centroid decomposition* T_C of T by removing a centroid v from T and selecting v as the root of the tree T_C. By removing v from T, we created disconnected subtrees of T. We recursively obtain their centroid decompositions and make their roots the children of v in T_C. See Figure 9 for an illustration. The resulting tree T_C has height $O(\log n)$. To construct a centroid decomposition, there exists a folklore $O(n \log n)$ algorithm and a linear-time algorithm [12]. During preprocessing, after computing a farthest-point Voronoi diagram in linear time, we also compute its centroid decomposition in linear time. Thus, the bound of Lemma 8 holds.

To efficiently find an edge or vertex x of a farthest-point Voronoi Diagrams FPVD(S), we traverse its centroid decomposition T_C. To "navigate" the decomposition, at a node v of T_C we need to decide which subtree contains x. Given a node v of T_C, which corresponds to a vertex of FPVD(S), an oracle $\mathcal{O}_x(v)$ selects the edge incident to v in FPVD(S) that lies on the path toward x. If $v = x$, the oracle reports success. Using the oracle we can efficiently identify x.

▶ **Lemma 11.** *Given an oracle $\mathcal{O}_x(v)$ and a centroid decomposition of* FPVD(S)*, we can find the vertex or edge x of* FPVD(S) *with $O(\log n)$ invocations of $\mathcal{O}_x(v)$.*

Finding the smallest enclosing disk of one set of points is now straightforward, as shown in the proof of the following theorem.

▶ **Theorem 12.** *Given a set of points S and their farthest-point Voronoi diagram in addition to its centroid decomposition, we can find the smallest enclosing disk of S in time $O(\log |S|)$.*

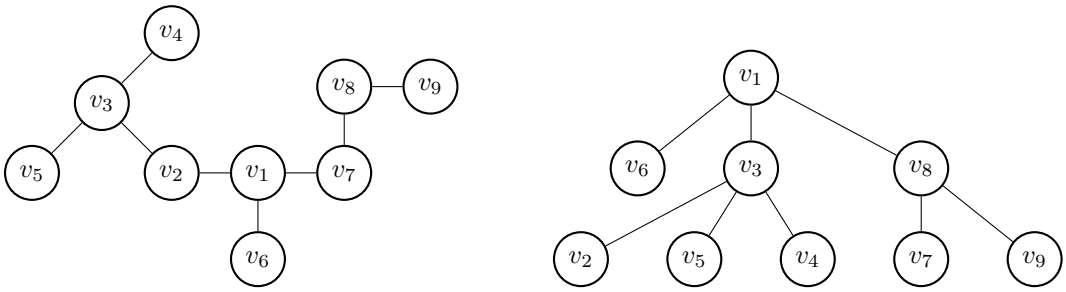


Figure 9 A tree T with centroid v_1 (left), and its centroid decomposition T_C (right).

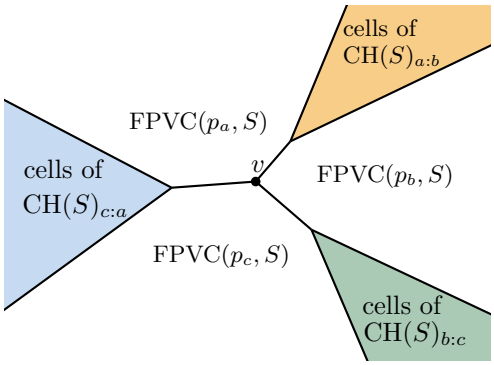


Figure 10 Conceptual sketch of a vertex v of $\mathrm{FPVD}(S)$, the incident edges and the relative locations of cells.

Proof. Let x be the vertex of $\mathrm{FPVD}(S)$ corresponding to the center z of $\mathrm{SED}(S)$, in case of three defining points, or be an edge of $\mathrm{FPVD}(S)$ containing z in case of two defining points. We use the procedure of Lemma 11 and provide an oracle $\mathcal{O}_x(v)$. The procedure implicitly maintains a subtree T of $\mathrm{FPVD}(S)$ and discards parts of T according to $\mathcal{O}_x(v)$. At any step T contains x. Let v be a vertex given to $\mathcal{O}_x(v)$ and p_a, p_b, p_c be its defining points. We first check if p_a, p_b, p_c form a non-obtuse triangle, in which case we report success, since no smaller disk can cover p_a, p_b, p_c.

Otherwise, we analyze the incident edges. The points that define $\mathrm{SED}(S)$ define an edge or vertex, and hence lie in exactly one of the sections $\mathrm{CH}(S)_{a:b}, \mathrm{CH}(S)_{b:c}, \mathrm{CH}(S)_{c:a}$.

In case the defining points lie in $\mathrm{CH}(S)_{a:b}$, let e be the corresponding incident edge defined by p_a and p_b, as illustrated by Figure 10. Take the point $s = v$, which lies on e. By analyzing the relative position of s relative to $\mathrm{mid}(p_a, p_b)$ by Lemma 10 we get an indication that $\mathrm{CH}(S)_{b+1:a-1}$ cannot contain defining points of $\mathrm{SED}(S)$.

In general, we analyze all incident edges as described and can conclude that two out of the three sections $\mathrm{CH}(S)_{a+1:b-1}, \mathrm{CH}(S)_{b+1:c-1}, \mathrm{CH}(S)_{c+1:a-1}$ cannot contain defining points of $\mathrm{SED}(S)$. Hence, we select the incident edge corresponding to the remaining section.

The points that define the vertex or edge x that the procedure of Lemma 11 returns are the defining points of $\mathrm{SED}(S)$. ◀

3.2 Determining the smallest enclosing disk for multiple sets

We develop a search procedure to identify the defining points contained by a given canonical section, which we invoke once for every canonical section. We first consider the overall goal of our search procedure, i.e. towards which parts of the diagrams we move. Then we show how we guide the search procedure.

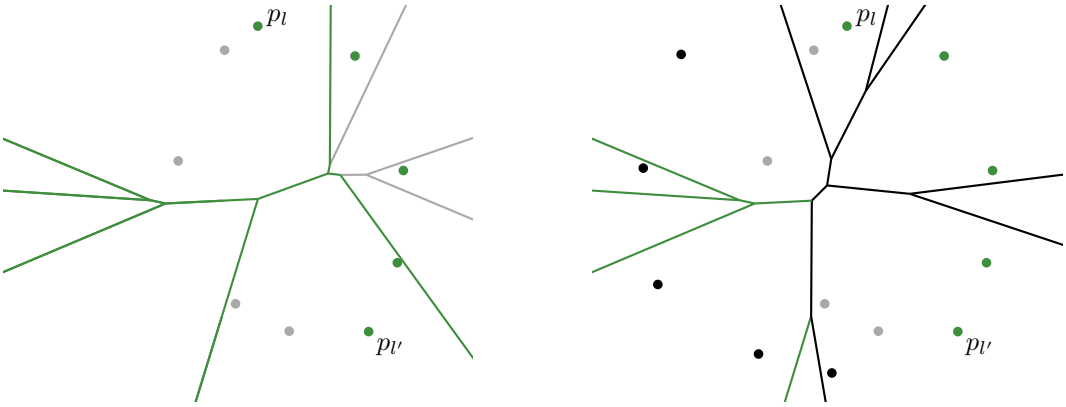


Figure 11 Diagram FPVD(S_i) (left) with a subtree induced by CH(S_i)$_{l:l'}$ (green), parts of that tree are also present in FPVD($S \cap Q$) (right).

Let G ("Goal") be the set of points that define SED($S \cap Q$). For every canonical section CH(S_i)$_{l:l'}$ of the convex hull we find an edge or vertex x of FPVD(S_i) that is adjacent to the cells of $G \cap \mathrm{CH}(S_i)_{l:l'}$. Whether x is an edge or vertex depends on the size of $G \cap \mathrm{CH}(S_i)_{l:l'}$.

$$x = \begin{cases} \text{an edge adjacent to } \mathrm{FPVC}(p_1, S_i), & \text{if } G \cap \mathrm{CH}(S_i)_{l:l'} = \{p_1\}, \\ \text{the edge defined by } p_1 \text{ and } p_2, & \text{if } G \cap \mathrm{CH}(S_i)_{l:l'} = \{p_1, p_2\}, \\ \text{the vertex defined by these three points,} & \text{if } |G \cap \mathrm{CH}(S_i)_{l:l'}| = 3. \end{cases}$$

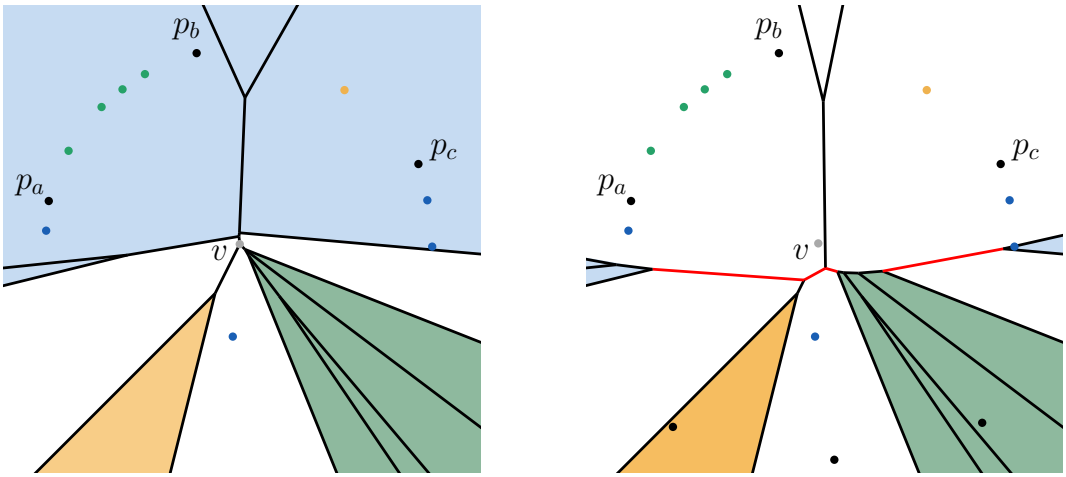
In case of $|G \cap \mathrm{CH}(S_i)_{l:l'}| = 0$, the procedure either detects that the canonical section does not contain any defining points and aborts or returns an edge defined by the point of CH(S_i)$_{l:l'}$ that admits the smallest enclosing disk having a point of CH(S_i)$_{l:l'}$ on its boundary. Once the search procedures for all $O(m)$ canonical sections is finished we gather all $O(m)$ defining points and determine the smallest enclosing disk with a linear-time algorithm.

To implement the search procedure, given a canonical section CH(S_i)$_{l:l'}$, we apply the procedure of Lemma 11 to FPVD(S_i) and provide an oracle $\mathcal{O}_x(v)$. Again, the procedure implicitly maintains a subtree T of FPVD(S_i) and discards parts of T according to the oracle $\mathcal{O}_x(v)$. We assume $|\,\mathrm{CH}(S_i)_{l:l'}| > 2$, otherwise we can immediately return the edge defined by the point(s) of CH(S_i)$_{l:l'}$.

Let v be the vertex of FPVD(S_i) given to $\mathcal{O}_x(v)$ and $p_a, p_b, p_c \in S_i$ be the defining points of v in clockwise order. By Observation 6 the section CH(S_i)$_{l:l'}$ induces a subtree T in FPVD(S_i) that contains v. Parts of T are also present in FPVD($S \cap Q$) as illustrated by Figure 11, since CH(S_i)$_{l:l'}$ is a canonical section. In time $O(m \log n)$ we decide if v is also a vertex of FPVD($S \cap Q$) by checking if no point is farther from v than p_a, p_b, p_c with queries in FPVD(S_1), …, FPVD(S_m).

If v is a vertex of FPVD($S \cap Q$), then this implies that p_a, p_b, p_c also define incident edges of v in FPVD($S \cap Q$). By comparing v to the pairwise midpoints of p_a, p_b, p_c by Lemma 10, we identify which section of CH(S_i)$_{a:b}$, CH(S_i)$_{b:c}$, CH(S_i)$_{c:a}$ may contain points of G as in the proof of Theorem 12. If this section is disjoint with CH(S_i)$_{l:l'}$ we abort the search procedure, otherwise we resolve the call to the oracle $\mathcal{O}_x(v)$ with the corresponding edge.

If v does not exist in FPVD($S \cap Q$), then the incident edges defined by p_a, p_b, p_c do not exist, as illustrated by Figure 12. In this case the cells FPVC($p_a, S \cap Q$), FPVC($p_b, S \cap Q$) and FPVC($p_c, S \cap Q$) clearly cannot meet at v, they are bounded by edges defined by points not in S_i (shown in red in the figure). Since CH($S \cap Q$) contains points of other



(a) Diagram FPVD(S_1) and points S_1.

(b) Diagram FPVD($S \cap Q$) and points $S \cap Q$.

Figure 12 Comparison of cells of points of S_1 in FPVD(S_1) and FPVD($S \cap Q$). The cells of the green points ($\mathrm{CH}(S_1)_{a+1:b-1}$), orange ($\mathrm{CH}(S_1)_{b+1:c-1}$) and blue points ($\mathrm{CH}(S_1)_{c+1:a-1}$) are shaded in their respective colors. In the latter diagram the red edges divide the cells of points of S_1.

sets, the canonical section may not include all points of S_i. In particular, it may not include all defining points p_a, p_b, p_c. We distinguish cases, depending on how many of the defining points and their neighbors are contained in $\mathrm{CH}(S_i)_{l:l'}$, i.e. on the number $k = |\{p_j \in \{p_a, p_b, p_c\} \mid \{p_{j-1}, p_j, p_{j+1}\} \subseteq \mathrm{CH}(S_i)_{l:l'}\}|$.

In case $k = 3$ we assume w.l.o.g. that the section $\mathrm{CH}(S_i)_{a-1:c+1}$ is contained in the canonical section, i.e. $\mathrm{CH}(S_i)_{a-1:c+1} \subseteq \mathrm{CH}(S_i)_{l:l'}$, otherwise we cyclically relabel p_a, p_b, p_c accordingly. We have the sequence

$$\mathrm{CH}(S_i)_{l:l'} = \{\underbrace{p_l, \ldots, p_{a-1}}_{\mathrm{CH}(S_i)_{l:a-1}}, p_a, \underbrace{p_{a+1}, \ldots, p_{b-1}}_{\mathrm{CH}(S_i)_{a+1:b-1}}, p_b, \underbrace{p_{b+1}, \ldots, p_{c-1}}_{\mathrm{CH}(S_i)_{b+1:c-1}}, p_c, \underbrace{p_{c+1}, \ldots, p_{l'}}_{\mathrm{CH}(S_i)_{c+1:l'}}\}.$$

For p_a, p_b, p_c define prev$(\cdot)$ and next$(\cdot)$ as all points that are before or after in the sequence relative to p_a, p_b, p_c respectively, i.e. $\mathrm{prev}(p_b) := \mathrm{CH}(S_i)_{l:b-1}$ and $\mathrm{next}(p_b) := \mathrm{CH}(S_i)_{b+1:l'}$. We will show that FPVC($p_a, S \cap Q$) has at least one *separating* edge e_a, i.e. an edge defined in conjunction with a point $p_{a'} \in \mathrm{CH}(S \cap Q)$ not lying in $\mathrm{CH}(S_i)_{l:l'}$. We define separating edges analogously for p_b, p_c. In Figure 12 separating edges were colored red.

▶ **Lemma 13.**

i) *The points p_a, p_b and p_c each have at least one separating edge.*

ii) *Removing a separating edge of $p_t \in \{p_a, p_b, p_c\}$ from* FPVD($S \cap Q$) *disconnects the cells of* next(p_t) *from those of* prev(p_t).

Proof. We prove the lemma for p_b. The other cases are analogous.

i). Consider the sequence of edges $e_1, \ldots, e_r$ bounding FPVC($p_b, S \cap Q$) in clockwise order. Let $q_1, \ldots, q_r \in S \cap Q$ be the points that define $e_1, \ldots, e_r$ respectively in conjunction with p_b. By Observation 4 the sequence $q_1, \ldots, q_r$ begins with points of next(p_b) and ends with points of prev(p_b), i.e. $q_1 = p_{b+1} \in \mathrm{next}(p_b)$ and $q_r = p_{b-1} \in \mathrm{prev}(p_b)$. Let q_j be the first point such that $q_j \notin \mathrm{next}(p_b)$. Such a point and corresponding edge has to exist, since otherwise v would also be a vertex in FPVD($S \cap Q$). Similarly, let $q_{j'}$ be the last point such that $q_{j'} \notin \mathrm{prev}(p_b)$. Note that $1 < j$ and $j' < r$ since $q_1 \in \mathrm{next}(p_b)$ and $q_r \in \mathrm{prev}(p_b)$. Now j, j' induce three parts of the sequence: A prefix $\{q_1, \ldots, q_{j-1}\} \subseteq \mathrm{next}\, p_b$, then $q_j, \ldots, q_{j'}$ and finally a suffix $\{q_{j'+1}, \ldots, q_r\} \subseteq \mathrm{prev}\, p_b$.

We show that $\mathrm{CH}(S_i)_{l:l'}$ contains none of the points $q_j, \ldots, q_{j'}$. Assume one of the points, say q_{j+1}, lies in $\mathrm{next}(p_b)$. Since $q_j \notin \mathrm{next}(p_b)$ and $q_1 \in \mathrm{next}(p_b)$ this would be a contradiction to Observation 4. Symmetrically $q_{j+1} \in \mathrm{prev}(p_b)$ cannot hold. Obviously $q_{j+1} \neq p_b$ holds as well, hence $q_{j+1} \notin \mathrm{CH}(S_i)_{l:l'}$. Thus, any edge e_s of $e_j, \ldots, e_{j'}$ is a separating edge of p_b.

ii). Let e_s be any edge of $e_j, \ldots, e_{j'}$. Removing e_s from $\mathrm{FPVD}(S \cap Q)$ creates two subtrees. The edges e_1 and e_r cannot lie in the same subtree, since we removed e_s which was part of the path from e_1 to e_r. Let T_1 denote the subtree containing e_1 and T_r denote the subtree containing e_r. The section $\mathrm{next}(p_b)$ by Observation 6 induces a subtree in $\mathrm{FPVD}(S \cap Q)$, since the point $p_{b+1} \in \mathrm{next}(p_b)$ defines e_1, this subtree is contained in T_1, Similarly, the induced subtree of $\mathrm{prev}(p_b)$ lies in T_r. We conclude that splitting $\mathrm{FPVD}(S \cap Q)$ on any of separating edge of p_b, disconnects the cells of $\mathrm{prev}(p_b)$ from those of $\mathrm{next}(p_b)$. ◀

Let e_a, e_b and e_c be separating edges of p_a, p_b and p_c, respectively, and let $p_{a'}, p_{b'}, p_{c'}$ be the other defining points, respectively, hence $p_{a'}, p_{b'}, p_{c'} \notin \mathrm{CH}(S_i)_{l:l'}$. By Lemma 10 an edge $e_j \in \{e_a, e_b, e_c\}$ with defining point $p_{j'}$ indicates $\mathrm{CH}(S \cap Q)_{j+1:j'-1} \cap G = \emptyset$ or $\mathrm{CH}(S \cap Q)_{j'+1:j-1} \cap G = \emptyset$ (otherwise we have found the center of $\mathrm{SED}(S \cap Q)$ and terminate the search procedure). Since $\mathrm{next}(p_j) \subseteq \mathrm{CH}(S \cap Q)_{j+1:j'-1}$ and $\mathrm{prev}(p_j) \subseteq \mathrm{CH}(S \cap Q)_{j'+1:j-1}$ it follows that either $\mathrm{next}(p_j) \cap G = \emptyset$ or $\mathrm{prev}(p_j) \cap G = \emptyset$.

We use $\leftarrow_j$ if by indication of e_j we know that $\mathrm{next}(p_j)$ cannot contain points of G and $\rightarrow_j$ in the other case. Now consider the possible outcomes if we analyze e_a, e_b, e_c this way:

- Case $\leftarrow_a, \leftarrow_b, \leftarrow_c$: Then $\mathrm{CH}(S_i)_{a+1:l'} \cap G = \emptyset$
- Case $\rightarrow_a, \leftarrow_b, \leftarrow_c$: Then $\mathrm{CH}(S_i)_{l:a-1} \cap G = \emptyset$ and $\mathrm{CH}(S_i)_{b+1:l'} \cap G = \emptyset$
- Case $\rightarrow_a, \rightarrow_b, \leftarrow_c$: Then $\mathrm{CH}(S_i)_{l:b-1} \cap G = \emptyset$ and $\mathrm{CH}(S_i)_{c+1:l'} \cap G = \emptyset$
- Case $\rightarrow_a, \rightarrow_b, \rightarrow_c$: Then $\mathrm{CH}(S_i)_{l:c-1} \cap G = \emptyset$

Unlisted cases, such as $\leftarrow_a \rightarrow_b \rightarrow_c$ would be a contradiction to Lemma 10, since $\leftarrow_a$ implies $\leftarrow_b$ and $\leftarrow_c$ due to $\mathrm{next}(p_a) \supseteq \mathrm{next}(p_b) \supseteq \mathrm{next}(p_c)$. For every case, we can conclude that only the respective "leftover" section $\mathrm{CH}(S_i)_{l:a}, \mathrm{CH}(S_i)_{a:b}, \mathrm{CH}(S_i)_{b:c}$ or $\mathrm{CH}(S_i)_{c:l'}$ possibly contains points of G. Note that there is exactly one corresponding edge of $\mathrm{FPVD}(S_i)$ incident to v for each of these sections, as illustrated by Figure 10. The edge defined by p_a and p_b corresponds to $\mathrm{CH}(S_i)_{a:b}$. The edge defined by p_a and p_c corresponds to $\mathrm{CH}(S_i)_{l:a}$ and $\mathrm{CH}(S_i)_{c:l'}$ since both sections are subsets of $\mathrm{CH}(S_i)_{c:a}$. Let e be the edge corresponding to the section we identified. Since x is incident to a cell of a point of G and hence in the subtree reachable with e, we select e as an answer for $\mathcal{O}_x(v)$.

For $k \leq 2$, we give the complete procedure in the full version [9]. For these cases, we must only determine and analyze k separating edges. Based on the indication given by Lemma 10 we identify which section $\mathrm{CH}(S_i)_{a:b}, \mathrm{CH}(S_i)_{b:c}, \mathrm{CH}(S_i)_{c:a}$ may contain points of $\mathrm{CH}(S_i)_{l:l'} \cap G$ and select the corresponding edge.

We have m canonical section, requiring $O(\log n)$ invocations of $\mathcal{O}_x(v)$ by Lemma 11. Every invocation involves analyzing up to three seperating edges. Hence, we obtain the following lemma. In the next section, we will then show how to find a seperating edge.

▶ **Lemma 14.** *By analyzing $O(m \log n)$ separating edges we can determine $\mathrm{SED}(S \cap Q)$.*

3.3 Finding separating edges

In this subsection we will give an $O(m \log n)$ procedure to find a point on a separating edge.

Consider the canonical sets of points $S_1, \ldots, S_m$, a point $p \in S_i$, and a vertex v of $\mathrm{FPVD}(S_i)$ defined by p. Additionally, we are given a canonical section $\mathrm{CH}(S_i)_{a:b}$ with $p \in \mathrm{CH}(S_i)_{a+1:b-1}$, i.e. the neighbors of p lie in $\mathrm{CH}(S_i)_{a:b}$. By assumption v does not exist in $\mathrm{FPVD}(S \cap Q)$. We will describe an $O(m \log n)$ time procedure to find a point s on an edge of $\mathrm{FPVD}(S \cap Q)$ defined by p and a point $p' \notin \mathrm{CH}(S_i)_{a:b}$, i.e. a point on a separating edge.

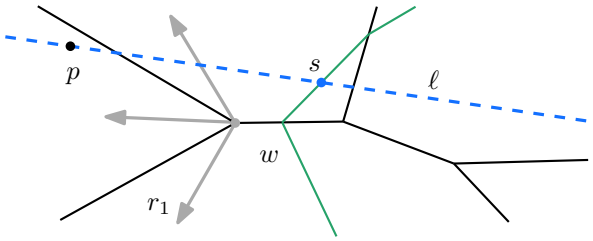


Figure 13 The point $p \in S_i$, the boundary of $\mathrm{FPVC}(p, S_j \cup \{p\})$ (green) and $\mathrm{FPVD}(S_j)$ (black).

Consider the ray r emanating from v and pointing opposite to p. We show that r intersects a separating edge of p.

▶ **Lemma 15.** *The ray r emanating from v and pointing opposite to p intersects an edge of $\mathrm{FPVC}(p, S \cap Q)$ defined by a point $p' \notin \mathrm{CH}(S_i)_{a:b}$.*

Proof. The half-infinite edges e_1, e_2 bounding $\mathrm{FPVC}(p, S \cap Q)$ are those of p and its neighbors contained in S_i. Let u, w be the vertices of $\mathrm{FPVD}(S \cap Q)$ that e_1 and e_2 are incident to respectively, then by convexity the line segment uw lies in $\mathrm{FPVC}(p, S \cap Q)$. By Observation 2 we have $r \subseteq \mathrm{FPVC}(p, S_i)$, this cell is also bounded by the half-infinite edges defined by p and its neighbors. Since the ray is fully contained by the induced halfspaces of e_1 and e_2 it intersects the line segment $uw \subseteq \mathrm{FPVC}(p, S \cap Q)$. The ray emanates from $v \notin \mathrm{FPVC}(p, S \cap Q)$. To intersect the line segment it must intersect an edge e of $\mathrm{FPVC}(p, S \cap Q)$. The other point p' that defines e cannot lie in S_i since the ray lies in the interior of $\mathrm{FPVC}(p, S_i)$. ◀

Thus, there is an intersection point of r and a separating edge. To determine this intersection point, observe that $\mathrm{FPVC}(p, S \cap Q) = \bigcap_{j=1}^{m} \mathrm{FPVD}(p, S_j \cup \{p\})$ holds by definition. We determine the intersection of r and the boundary of $\mathrm{FPVD}(p, S_i \cup \{p\})$ for every $S_j \neq S_i$, then the most restrictive intersection point, i.e. farthest from v, corresponds to a point on an edge of $\mathrm{FPVC}(p, S \cap Q)$. We extend r to infinity in both directions to get the line $\ell \supset r$.

▶ **Lemma 16.** *For every set S_j in time $O(\log n)$ we can find the intersection of ℓ and an edge e of $\mathrm{FPVC}(p, S_j \cup \{p\})$ and its defining point p'.*

Proof. Let $s = \ell \cap e$ be the intersection point. Observe that $s \in \mathrm{FPVC}(p', S_j \cup \{p\}) \subseteq \mathrm{FPVC}(p', S_j)$, thus we identify the cell of $\mathrm{FPVD}(S_j)$ containing the not explicitly known point s. We use the search procedure of Lemma 11 to efficiently find an arbitrary edge of $\mathrm{FPVC}(p', S_j)$ with a technique inspired by [5]. In every step we ensure that the implicitly maintained subtree of $\mathrm{FPVD}(S_j)$ contains an edge of $\mathrm{FPVC}(p', S_j)$.

Let w be vertex given to the oracle $\mathcal{O}_x(w)$. By Observation 5 the rays r_1, r_2, r_3 emanating from w and pointing opposite to the defining points q_1, q_2, q_3 of w divide the plane in three regions having one corresponding incident edge each as illustrated by Figure 13. The region containing s also contains a bounding edge x of $\mathrm{FPVC}(p', S_j)$.

To decide which region contains s, we may have to determine the location of s relative to an intersection of ℓ and a ray r_k of r_1, r_2, r_3. Observe that s by definition splits ℓ into two parts, the part where p is the farthest point extending along r, and the part where p is closer than another point. Thus, to infer the location of s relative to $\ell \cap r_k$ we must only compare the distance of p and q_k and choose the region we reach by moving along r from $\ell \cap r_k$ exactly when p is closer.

The procedure of Lemma 11 invokes $\mathcal{O}_x(w)$ times $O(\log n)$, we resolve every invocation in time $O(1)$. Once a single edge $\mathrm{FPVD}(S_j)$ remains, the procedure of Lemma 11 returns the edge. In constant time we decide which defining point is p'. ◀

Thus, we can find m intersection points $s_1, \ldots, s_m$ and defining points $p_1, \ldots, p_m$ in time $O(m \log n)$. In time $O(m)$ we identify the point s_j farthest along r and return (s_j, p, p_j).

3.4 Putting things together

To determine $\mathrm{SED}(S \cap Q)$ we derive $m = O(\log n)$ canonical sets and canonical sections of $\mathrm{CH}(S \cap Q)$ in time $O(\log^3 n)$ using our search data structure as in Lemma 8. The search procedure for every canonical set introduced in Subsection 3.2 has $O(\log n)$ steps. Each step involves finding up to three separating edges. With the procedure of Subsection 3.3 we find a separating edge in time $O(m \log n)$. Finally, we determine $\mathrm{SED}(S \cap Q)$ in time $O(m)$. Overall, we determine $\mathrm{SED}(S \cap Q)$ in time $O(m^2 \log^2 n)$.

▶ **Theorem 17.** *Given sets $S_1, \ldots, S_m$ with $|S_i| = O(n)$, the diagrams $\mathrm{FPVD}(S_1), \ldots, \mathrm{FPVD}(S_m)$, their centroid decompositions and the canonical sections of the convex hull of $\bigcup_{i=1}^m S_i$, in time $O(m^2 \log^2 n)$ we can find $\mathrm{SED}(\bigcup_{i=1}^m S_i)$, assuming the convex hull consists of $O(m)$ canonical sections.*

▶ **Theorem 18.** *Given a constant number of sets having a constant number of canonical sections, we can find the smallest enclosing disk in time $O(\log^2 n)$.*

In the context of range-aggregate queries, the number of sets is $m = O(\log n)$ and the convex hull consists of $O(m)$ canonical sections by Lemma 1. This yields the following theorem.

▶ **Theorem 19.** *Let S be a set of points of size n in the plane. With $O(n \log^2 n)$ preprocessing time and storage we can answer orthogonal range-aggregate queries for the smallest enclosing disk in time $O(\log^4 n)$.*

We note that the search procedure is versatile enough to find other points on the graph $\mathrm{FPVD}(S \cap Q)$, provided we can determine which subtree contains them, given an edge.

4 Randomized approaches

A well known randomized algorithm that determines $\mathrm{SED}(S)$ given a set of n points S in expected linear time is Welzl's algorithm [26]. A natural idea is to try to generalize this algorithm to sets of disjoint convex polygons. In Subsection 4.1 we present a counter example that shows that a direct adaptation unfortunately does not work. However, we can show that we can plug in our implicit searching procedure from the previous section into Eppstein's randomized dynamic programming framework, see Subsection 4.2, to answer smallest enclosing disk queries in expected $O(\log^{5/2} n \log \log n)$ time instead.

4.1 An attempt at randomized incremental construction

Welzl's algorithm [26] builds upon three central properties:

- $\mathrm{SED}(S)$ is defined by up to three points.
- By testing $p \notin \mathrm{SED}(S \setminus \{p\})$, we can decide whether any point $p \in S$ is a defining point.
- The probability that a random point $p \in S$ is a defining point is $O(\frac{1}{n})$.

This allows for a simple procedure: Consider the points $p_1, \ldots, p_n \in S$ in random order and maintain an intermediate disk D_i at every step. When considering p_{i+1}, in case of $p_{i+1} \in D_i$ set $D_{i+1} := D_i$, otherwise p_{i+1} is a defining point and lies on the boundary of D_{i+1}. Recurse to construct $D_{i+1} := \mathrm{MD}(S', R)$ with $S' = \{p_1, \ldots, p_i\}$ and $R = \{p_{i+1}\}$. The disk $\mathrm{MD}(S', R)$ denotes the smallest disk covering the points $S' \cup R$, while having all points of R on its

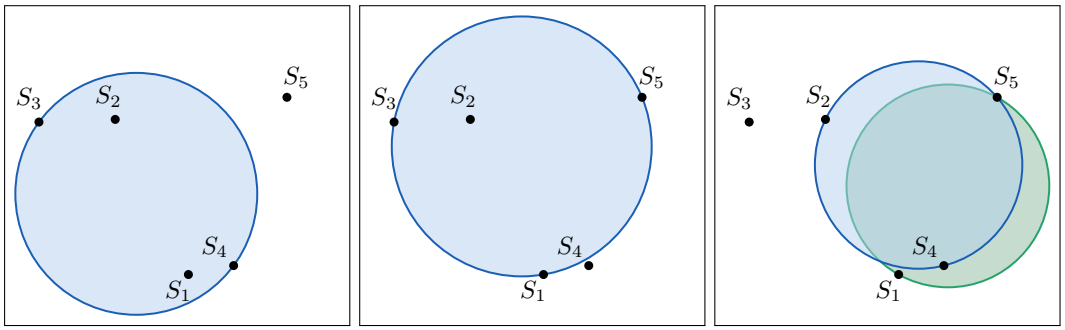


Figure 14 Example showing that not any intermediate disk is defined. From left to right: $\mathrm{SMD}(\{S_1, \ldots, S_4\}, \emptyset)$, $\mathrm{SMD}(\{S_1, S_2, S_3\}, \{S_5\})$, $\mathrm{SMD}(\{S_1, S_2\}, \{S_4, S_5\})$.

boundary. Any recursive subproblem $\mathrm{MD}(S', R)$ can be solved with the same strategy. For $|R| = 3$ the disk is uniquely defined. The probability that $p_{i+1} \notin D_i$ holds is $O(\frac{1}{i})$, leading to $O(n)$ expected runtime when solving $\mathrm{MD}(S, \emptyset) = \mathrm{SED}(S)$.

Given canonical sets $S_1, \ldots, S_m$ we can test $S_i \subseteq D$ for any disk D in time $O(\log n)$, by checking if D contains the point that is farthest from its center. In case $S_{i+1} \not\subseteq \mathrm{SED}(S_1 \cup \cdots \cup S_i)$ we can conclude S_{i+1} contains at least one defining point. Thus, a natural idea is to lift Welzl's algorithm to operate on sets, instead of single points, resulting in procedure `SetMiniDisk` $(\mathcal{S}, \mathcal{R})$, which computes $\mathrm{SMD}(\mathcal{S}, \mathcal{R})$. The set $\mathcal{R}$ holds up to three sets of $S_1, \ldots, S_m$ that contain defining points. Using our procedure we can solve base cases involving three sets in time $O(\log^2 n)$ by Theorem 18.

Procedure SetMiniDisk$(\mathcal{S}, \mathcal{R})$.

if $|\mathcal{R}| = 3$ **or** $|\mathcal{S}| = 0$ **then**
  return $\mathrm{SED}(\bigcup_{S_i \in \mathcal{R}} S_i)$;
Choose a set $S \in \mathcal{S}$ uniformly at random;
$D \leftarrow$ `SetMiniDisk`$(\mathcal{S} \setminus \{S\}, \mathcal{R})$;
if $S \notin D$ **then**
  $D \leftarrow$ `SetMiniDisk`$(\mathcal{S} \setminus \{S\}, \mathcal{R} \cup \{S\})$;
return D;

Unfortunately, there are instances where this procedure encounters undefined intermediate disks. Figure 14 shows an example, where neither disk in the last frame can cover S_1 and S_2 at the same time. Refer to the full version [9] for a more detailed description of the procedure and the counter example.

4.2 Using Eppstein's dynamic programming framework

Eppstein presented a framework for linear programming in the intersection of k polyhedra in $\mathbb{R}^3$. The dynamic programming based framework determines the optimum in expected time $O(k \log k \log n + \sqrt{k} \log k \log^3 n)$ [15]. To adapt the framework, we test if a set S_i contains a boundary point of $\mathrm{SED}(S \cap Q)$, by checking whether $S_i \not\subseteq \mathrm{SED}(\bigcup_{j \neq i} S_j)$ holds.

▶ **Lemma 20.** *If* $S_i \not\subseteq \mathrm{SED}(\bigcup_{j \neq i} S_j)$ *then* S_i *contains a boundary point of* $\mathrm{SED}(\bigcup_{j=1}^{m} S_j)$.

The probability is $O(\frac{1}{m})$, when $S_1 \ldots, S_m$ are ordered randomly. We compute $\mathrm{SED}(\bigcup_{j\neq i} S_j)$ recursively, by leaving out single sets and computing the smallest enclosing disk for the remaining sets. We give the complete algorithm in the full version [9]. A base case involves three sets, which we solve with our procedure in time $O(\log^2 n)$ by Theorem 18. The number of checks, such as $S_i \nsubseteq \mathrm{SED}(\bigcup_{j\neq i} S_j)$ is $O(m \log m)$ and the number of solved base cases is $O(\sqrt{m} \log m)$ as shown by [15]. Thus, we get the following theorem.

▶ **Theorem 21.** *Given sets $S_1, \ldots, S_m$ with $|S_i| = O(n)$, the diagrams $\mathrm{FPVD}(S_1), \ldots, \mathrm{FPVD}(S_m)$ and their centroid decompositions, we can find $\mathrm{SED}(\bigcup_{i=1}^{m} S_i)$ in expected time $O(m \log m \log n + \sqrt{m} \log m \log^2 n)$, assuming the convex hull of every triplet of sets (S_i, S_j, S_k) consists of $O(1)$ canonical sections.*

In the context of range-aggregate queries, the number of sets is $m = O(\log n)$ and the convex hull of every triplet has $O(1)$ canonical sections. This leads to the following theorem.

▶ **Theorem 22.** *Let S be a set of points of size n in the plane. With $O(n \log^2 n)$ preprocessing time and storage we can answer orthogonal range-aggregate queries for the smallest enclosing disk in expected time $O(\log^{\frac{5}{2}} n \log \log n)$.*

References

1. Mikkel Abrahamsen, Mark de Berg, Kevin Buchin, Mehran Mehr, and Ali D. Mehrabi. Range-Clustering Queries. In Boris Aronov and Matthew J. Katz, editors, *33rd International Symposium on Computational Geometry (SoCG 2017)*, volume 77 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 5:1–5:16, Dagstuhl, Germany, 2017. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.SoCG.2017.5.
2. Peyman Afshani, Pingan Cheng, Aniket Basu Roy, and Zhewei Wei. On range summary queries. In Kousha Etessami, Uriel Feige, and Gabriele Puppis, editors, *50th International Colloquium on Automata, Languages, and Programming, ICALP 2023, Paderborn, Germany, July 10-14, 2023*, LIPIcs, pages 7:1–7:17. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2023. doi:10.4230/LIPIcs.ICALP.2023.7.
3. Pankaj K. Agarwal and Cecilia Magdalena Procopiuc. Exact and approximation algorithms for clustering. *Algorithmica*, 33(2):201–226, 2002. doi:10.1007/S00453-001-0110-Y.
4. Alok Aggarwal, Leonidas J. Guibas, James Saxe, and Peter W. Shor. A linear-time algorithm for computing the voronoi diagram of a convex polygon. *Discrete & Computational Geometry*, 4(6):591–604, December 1989. doi:10.1007/BF02187749.
5. Boris Aronov, Prosenjit Bose, Erik D. Demaine, Joachim Gudmundsson, John Iacono, Stefan Langerman, and Michiel Smid. Data Structures for Halfplane Proximity Queries and Incremental Voronoi Diagrams. *Algorithmica*, 80(11):3316–3334, November 2018. doi:10.1007/s00453-017-0389-y.
6. Franz Aurenhammer, Rolf Klein, and Der-tsai Lee. *Voronoi Diagrams And Delaunay Triangulations*. World Scientific Publishing Company, June 2013. doi:10.1142/8685.
7. Mihai Badoiu, Sariel Har-Peled, and Piotr Indyk. Approximate clustering via core-sets. In John H. Reif, editor, *Proceedings on 34th Annual ACM Symposium on Theory of Computing, May 19-21, 2002, Montréal, Québec, Canada*, pages 250–257. ACM, 2002. doi:10.1145/509907.509947.
8. Peter Brass, Christian Knauer, Chan-Su Shin, Michiel H. M. Smid, and Ivo Vigan. Range-Aggregate Queries for Geometric Extent Problems. In Anthony Wirth, editor, *Nineteenth Computing: The Australasian Theory Symposium, CATS 2013, Adelaide, Australia, February 2013*, volume 141 of *CRPIT*, pages 3–10. Australian Computer Society, 2013. URL: https://dl.acm.org/doi/10.5555/2525519.2525520.

9 Kevin Buchin, Mark Joachim Krallmann, and Frank Staals. Smallest Enclosing Disk Queries Using Farthest-Point Voronoi Diagrams, May 2026. doi:10.48550/arXiv.2605.00743.

10 Bernard Chazelle. Lower bounds for orthogonal range searching II. the arithmetic model. *J. ACM*, 37(3):439–463, 1990. doi:10.1145/79147.79149.

11 Mark De Berg, Otfried Cheong, Marc Van Kreveld, and Mark Overmars. *Computational Geometry: Algorithms and Applications*. Springer, Berlin, Heidelberg, 2008. doi:10.1007/978-3-540-77974-2.

12 Davide Della Giustina, Nicola Prezza, and Rossano Venturini. A New Linear-Time Algorithm for Centroid Decomposition. In Nieves R. Brisaboa and Simon J. Puglisi, editors, *String Processing and Information Retrieval*, pages 274–282, Cham, 2019. Springer International Publishing. doi:10.1007/978-3-030-32686-9_20.

13 Herbert Edelsbrunner and Ernst Peter Mücke. Simulation of simplicity: A technique to cope with degenerate cases in geometric algorithms. *ACM Trans. Graph.*, 9(1):66–104, January 1990. doi:10.1145/77635.77639.

14 Jack Elzinga and Donald W. Hearn. Geometrical solutions for some minimax location problems. *Transportation Science*, 6(4):379–394, 1972. doi:10.1287/trsc.6.4.379.

15 David Eppstein. Dynamic Three-Dimensional Linear Programming. *ORSA Journal on Computing*, 4(4):360–368, November 1992. doi:10.1287/ijoc.4.4.360.

16 Erwin Glazenburg and Frank Staals. On strictly output-sensitive color frequency reporting. In Jakub Kozik and Alexander Wolff, editors, *SOFSEM 2026: Theory and Practice of Computer Science - 51st International Conference on Current Trends in Theory and Practice of Computer Science, SOFSEM 2026, Kraków, Poland, February 9-13, 2026, Proceedings*, Lecture Notes in Computer Science, pages 246–259. Springer, 2026. doi:10.1007/978-3-032-17801-5_18.

17 Sariel Har-Peled and Soham Mazumdar. On coresets for k-means and k-median clustering. In *Proceedings of the Thirty-Sixth Annual ACM Symposium on Theory of Computing*, STOC '04, pages 291–300, New York, NY, USA, June 2004. Association for Computing Machinery. doi:10.1145/1007352.1007400.

18 Ziyun Huang and Jinhui Xu. In-Range Farthest Point Queries and Related Problem in High Dimensions. In Mikołaj Bojańczyk, Emanuela Merelli, and David P. Woodruff, editors, *49th International Colloquium on Automata, Languages, and Programming (ICALP 2022)*, volume 229 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 75:1–75:21, Dagstuhl, Germany, 2022. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.ICALP.2022.75.

19 Camille Jordan. Sur les assemblages de lignes. *Journal für die reine und angewandte Mathematik*, 70:185–190, 1869. doi:10.1515/9783112389409-014.

20 Sankalp Khare, Jatin Agarwal, Nadeem Moidu, and Kannan Srinathan. Improved Bounds for Smallest Enclosing Disk Range Queries. In *Proceedings of the 26th Canadian Conference on Computational Geometry, CCCG 2014, Halifax, Nova Scotia, Canada, 2014*. Carleton University, Ottawa, Canada, 2014. URL: http://www.cccg.ca/proceedings/2014/papers/paper42.pdf.

21 Nimrod Megiddo. Linear-Time Algorithms for Linear Programming in $\boldsymbol{R}^3$ and Related Problems. *SIAM Journal on Computing*, 12(4):759–776, November 1983. doi:10.1137/0212052.

22 Yakov Nekrich. Data structures for approximate orthogonal range counting. In Yingfei Dong, Ding-Zhu Du, and Oscar H. Ibarra, editors, *Algorithms and Computation, 20th International Symposium, ISAAC 2009, Honolulu, Hawaii, USA, December 16-18, 2009. Proceedings*, Lecture Notes in Computer Science, pages 183–192. Springer, 2009. doi:10.1007/978-3-642-10631-6_20.

23 Yakov Nekrich and Michiel H. M. Smid. Approximating range-aggregate queries using coresets. In *Proceedings of the 22nd Annual Canadian Conference on Computational Geometry, Winnipeg, Manitoba, Canada, August 9-11, 2010*, pages 253–256, 2010. URL: https://cccg.ca/proceedings/2010/paper67.pdf.

24 Eunjin Oh and Hee-Kap Ahn. Approximate Range Queries for Clustering. In Bettina Speckmann and Csaba D. Tóth, editors, *34th International Symposium on Computational Geometry (SoCG 2018)*, volume 99 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 62:1–62:14, Dagstuhl, Germany, 2018. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.SoCG.2018.62.

25 Michael Ian Shamos and Dan Hoey. Closest-point problems. In *16th Annual Symposium on Foundations of Computer Science (Sfcs 1975)*, pages 151–162, October 1975. doi:10.1109/SFCS.1975.8.

26 Emo Welzl. Smallest enclosing disks (balls and ellipsoids). In Hermann Maurer, editor, *New Results and New Trends in Computer Science*, pages 359–370, Berlin, Heidelberg, 1991. Springer. doi:10.1007/BFb0038202.