

# Counting Unit Circular Arc Intersections

Haitao Wang 

Kahlert School of Computing, University of Utah, Salt Lake City, UT, USA

---

## Abstract

Given a set of  $n$  circular arcs of the same radius in the plane, we consider the problem of computing the number of intersections among the arcs. The problem was studied before and the previously best algorithm solves the problem in  $O(n^{4/3+\epsilon})$  time [Agarwal, Pellegrini, and Sharir, SIAM J. Comput., 1993], for any constant  $\epsilon > 0$ . No progress has been made on the problem for more than 30 years. We present a new algorithm of  $O(n^{4/3} \log^{16/3} n)$  time and improve it to  $O(n^{1+\epsilon} + K^{1/3} n^{2/3} (\frac{n^2}{n+K})^\epsilon \log^{16/3} n)$  time for small  $K$ , where  $K$  is the number of intersections of all arcs.

**2012 ACM Subject Classification** Theory of computation  $\rightarrow$  Computational geometry; Theory of computation  $\rightarrow$  Design and analysis of algorithms

**Keywords and phrases** circular arc intersections, unit circles, arrangements, cuttings, segment intersections

**Digital Object Identifier** 10.4230/LIPIcs.STACS.2026.81

**Related Version** *Full Version:* <https://arxiv.org/abs/2602.01074>

## 1 Introduction

Given a set of  $n$  circular arcs of the same radius in the plane, we consider the problem of computing the number of intersections among the arcs; here we count the number of intersecting points (i.e., if two arcs intersect at two points, then they are counted twice). Agarwal, Pellegrini, and Sharir [3] previously solved the problem in  $O(n^{4/3+\epsilon})$  time; throughout the paper, we let  $\epsilon$  denote an arbitrarily small positive constant. On the negative side, Erickson's results [11] show that  $\Omega(n^{4/3})$  is a lower bound for the problem in a so-called partition algorithm model. No progress has been made on the problem for more than 30 years. In this paper, we present a new algorithm of  $O(n^{4/3} \log^{16/3} n)$  time and thus improve the previous result in [3]. Although the improvement looks minor, a factor of  $O(n^\epsilon)$ , it is significant from the following perspective: it reduces the gap between the upper and lower bounds from  $\text{poly}(n)$  to  $\text{poly}(\log n)$ , where  $\text{poly}(\cdot)$  is a polynomial function. Further, for small  $K$ , where  $K$  is the number of intersections of all arcs, we improve the algorithm to  $O(n^{1+\epsilon} + K^{1/3} n^{2/3} (\frac{n^2}{n+K})^\epsilon \log^{16/3} n)$  time. Note that this matches the above  $O(n^{4/3} \log^{16/3} n)$  complexity in the worst case when  $K = \Theta(n^2)$ . If  $K = O(n^{2-\delta})$  for any  $\delta > 0$ , then the time complexity is  $o(n^{4/3})$ .

In addition, our results can be used to solve the following *bichromatic problem* (with the same time complexity): Given a set of red circular arcs and another set of blue circular arcs such that all arcs (of both colors) have the same radius, compute the number of red-blue intersections.

**Related work.** If the circular arcs have different radii, Agarwal, Pellegrini, and Sharir [3] gave an  $O(n^{3/2+\epsilon})$  time algorithm to compute the number of intersections of all arcs.

For computing the number of intersections among a set of  $n$  circles of the same radius, Katz and Sharir [13] gave an algorithm of  $O(n^{4/3} \log n)$  time and the randomized algorithm of Agarwal, Aronov, Sharir, and Suri [2] can solve it in  $O(n^{4/3} \log^{2/3} n)$  expected time. Recently, Wang [16] derived an  $O(n^{4/3})$  time algorithm for it, matching the  $\Omega(n^{4/3})$  lower bound [11].



© Haitao Wang;

licensed under Creative Commons License CC-BY 4.0

43rd International Symposium on Theoretical Aspects of Computer Science (STACS 2026).

Editors: Meena Mahajan, Florin Manea, Annabelle McIver, and Nguyễn Kim Thăng

Article No. 81; pp. 81:1–81:18



Leibniz International Proceedings in Informatics

LIPICs Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany



If the circles have different radii, then the problem is solvable in  $O(n^{3/2+\epsilon})$  time [3]. Recently, Chan, Cheng, and Zheng [7] extended the result and gave an  $O(n^{3/2+\epsilon})$ -time algorithm to compute the number of intersections among a set of  $n$  algebraic arcs of constant description complexity in the plane.

The case where all arcs are line segments has been extensively studied. First of all, Erickson's  $\Omega(n^{4/3})$ -time lower bound [11] still applies. On the positive side, Chazelle [9] gave an  $O(n^{1.695})$  time algorithm for the problem. Guibas, Overmars, and Sharir [12] developed a randomized algorithm of  $O(n^{4/3+\epsilon})$  expected time. Agarwal [1] solved the problem in  $O(n^{4/3} \log^{1.78} n)$  time and Chazelle [10] further improved the algorithm to  $O(n^{4/3} (\log n)^{1/3})$  time. For small  $K$ , where  $K$  is the number of all segment intersections, Pellegrini [15] gave an algorithm of  $O(n^{1+\epsilon} + K^{1/3} n^{2/3+\epsilon})$  time. De Berg and Schwarzkopf [6] presented a randomized algorithm of  $O(n \log n \log K + K^{1/3} n^{2/3} \log^{1/3} n)$  expected time. Recently, Chan and Zheng [8] solved the problem in  $O(n^{4/3})$  time, matching the  $\Omega(n^{4/3})$  lower bound [11].

With Chan and Zheng's new result [8] and our techniques, we can show that the segment case problem is solvable in  $O(n^{1+\epsilon} + K^{1/3} n^{2/3} (\frac{n^2}{n+K})^\epsilon)$  time, or in  $O(n \log n \log K + K^{1/3} n^{2/3})$  expected time with the randomized techniques of de Berg and Schwarzkopf [6].

One essential difference between the segment case and the circular arc case is that two segments have at most one intersection while two circular arcs may have two intersections. This difference makes the technique of Chan and Zheng's new result [8] not applicable to the arc case. Indeed, this difference also makes the circular arc case more challenging.

In addition to a polynomial-time improvement over the 30-year old previous work [3] on a fundamental problem, our contribution also lies on many new geometric observations and techniques on the unit circular arcs. These new techniques may find applications in solving other related problems on unit circular arcs as well.

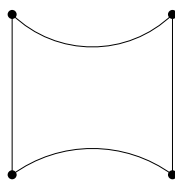
**Outline.** After introducing some concepts in Section 2, we present our algorithm in Section 3 for computing the number of intersections among circular arcs. Section 4 concludes and also demonstrates that our techniques may be extended to solve other variations, such as the bichromatic version and the segment case. Due to the space limit, some proofs are omitted but can be found in the full paper.

## 2 Preliminaries

We introduce some notation that will be used later. For two points  $a$  and  $b$  in the plane, we use  $\overline{ab}$  to denote the segment with endpoints  $a$  and  $b$ ; we call  $\overline{ab}$  the *defining segment* of  $a$  (resp.,  $b$ ). For any compact region  $A$  in the plane, we use  $\partial A$  to denote the boundary of  $A$ .

**Cuttings.** Let  $H$  be a set of  $n$  lines in the plane. For a compact region  $A$  in the plane, we use  $H_A$  to denote the subset of lines of  $H$  that intersect the interior of  $A$  (we also say that these lines *cross*  $A$ ). A *cutting* is a collection  $\Xi$  of closed cells (each of which is a triangle) with disjoint interiors, which together cover the entire plane [10, 14]. The *size* of  $\Xi$  is the number of cells in  $\Xi$ . For a parameter  $r$  with  $1 \leq r \leq n$ , a  $(1/r)$ -*cutting* for  $H$  is a cutting  $\Xi$  satisfying  $|H_\sigma| \leq n/r$  for every cell  $\sigma \in \Xi$ .

We say that a cutting  $\Xi'$  *c-refines* a cutting  $\Xi$  if every cell of  $\Xi'$  is contained in a single cell of  $\Xi$  and each cell of  $\Xi$  contains at most  $c$  cells of  $\Xi'$ . Let  $\Xi_0, \Xi_1, \dots, \Xi_k$  be a sequence of cuttings for  $H$  such that  $\Xi_0$  is the entire plane, and every  $\Xi_i$  is a  $(1/\rho^i)$ -cutting of size  $O(\rho^{2i})$  which  $c$ -refines  $\Xi_{i-1}$ , for two constants  $c$  and  $\rho > 1$ . To make  $\Xi_k$  a  $(1/r)$ -cutting, we set  $k = \lceil \log_\rho r \rceil$ . The above sequence of cuttings is called a *hierarchical*  $(1/r)$ -*cutting* for  $H$ . If a cell  $\sigma \in \Xi_{j-1}$  contains a cell  $\sigma' \in \Xi_j$ , we say that  $\sigma$  is the *parent* of  $\sigma'$  and  $\sigma'$  is a *child* of  $\sigma$ .



■ **Figure 1** Illustrating a pseudo-trapezoid. The top (resp., bottom) side is a circular arc that is part of an input arc. The left (resp., right) side is a vertical segment.

For any  $1 \leq r \leq n$ , a hierarchical  $(1/r)$ -cutting of size  $O(r^2)$  for  $H$  (together with the sets  $H_\sigma$  for every cell  $\sigma$  of  $\Xi_i$  for all  $i = 0, 1, \dots, k$ ) can be computed in  $O(nr)$  time [10]. Note that this implies that the total size of  $H_\sigma$  for all cells  $\sigma \in \Xi_i$ ,  $0 \leq i \leq k$ , is  $O(nr)$ .

As indicated by Agarwal and Sharir [4], similar results on cuttings for other more general curves in the plane (e.g., circles or circular arcs of different radii, pseudo-lines, line segments) can also be obtained by extending Chazelle's algorithm [10]; Wang [16] provided algorithm details for that. For our purpose, if  $H$  is a set of  $n$  circular arcs (not necessarily with the same radius) in the plane, then we can also define cuttings for  $H$  in the same way as above. A difference is that a cell  $\sigma$  of a cutting becomes a *pseudo-trapezoid* (more specifically, each pseudo-trapezoid in general has four sides, with the left/right side as a vertical line segment and the top/bottom side as part of an input circular arc; e.g., see Fig. 1). The following result, which will be used later, has been obtained in [16].

► **Theorem 1** ([16]). *Suppose  $H$  is a set of  $n$  circular arcs (or line segments) in the plane and  $K$  is the number of intersections of the arcs of  $H$ . For any  $r \leq n$ , a hierarchical  $(1/r)$ -cutting of size  $O(r^2)$  for  $H$  (together with the subsets  $H_\sigma$  for every cell  $\sigma$  of  $\Xi_i$  for all  $0 \leq i \leq k$ ) can be computed in  $O(nr)$  time; more specifically, the size of the cutting is  $O(r^{1+\epsilon} + K \cdot r^2/n^2)$  and the runtime of the algorithm is  $O(nr^\epsilon + K \cdot r/n)$ , for any  $\epsilon > 0$ .*

Cuttings are one of our main tools. Although cuttings is a standard technique, the novelty of our approach lies in how to combine cuttings with the newly discovered geometric observations and techniques on unit circular arcs.

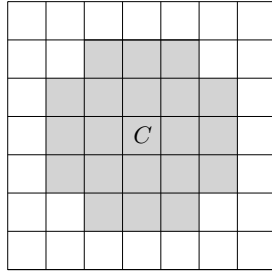
### 3 The algorithm

We present an  $O(n^{4/3} \log^{16/3} n)$ -time algorithm for computing the number of intersections of  $n$  circular arcs of the same radius in the plane. At the very end of this section, we reduce the time to  $O(K^{1/3} n^{2/3} (\frac{n^2}{n+K})^\epsilon \log^{16/3} n)$ , where  $K$  is the number of intersections of all arcs.

Let  $S$  be a set of  $n$  circular arcs of the same radius in the plane. Without loss of generality, we assume that the radius is 1, and each arc is thus a *unit circular arc*. Our goal is to compute the number of intersections of all arcs of  $S$ . In the following, unless otherwise stated, a circular arc refers to a unit circular arc.

We call a disk a *unit disk* if its radius is 1; the boundary of a unit disk is a *unit circle*. For each arc  $s$ , the circle that contains it is called its *underlying circle* (the disk bounded by the circle is called the *underlying disk*), and the center of the circle is also called the *center* of  $s$ . We use  $\alpha(s)$  and  $D(s)$  to denote the underlying circle and disk of  $s$ , respectively. Let  $P$  denote the set of the centers of the arcs of  $S$ . For any region  $A$  in the plane, let  $P(A)$  denote the subset of points of  $P$  in  $A$ , i.e.,  $P(A) = P \cap A$ .

With respect to  $P$ , Wang [16] gave an algorithm to compute a set  $\mathcal{C}$  of  $O(n)$  axis-parallel square cells in the plane whose interiors are pairwise disjoint and whose union covers all the points of  $P$ , with the following properties: (1) Each cell has side length  $1/\sqrt{2}$ . (2) Every two



■ **Figure 2** The set  $N(C)$  is a subset of the grey cells. Because the side-length of each cell is  $1/\sqrt{2}$ , the length between any point in  $C$  and any point not in a grey cell must be larger than 1.

cells are separated by an axis-parallel line. (3) For a unit disk  $D_p$  with center  $p$ , if  $p$  is not in any cell of  $\mathcal{C}$ , then  $D_p \cap P = \emptyset$ . (4) Each cell  $C$  of  $\mathcal{C}$  is associated with a subset  $N(C)$  of  $O(1)$  cells of  $\mathcal{C}$ , such that for any disk  $D$  with center in  $C$ , every point of  $P \cap D$  is in one of the cells of  $N(C)$ ; e.g., see Fig. 2. (5) Each cell  $C'$  of  $\mathcal{C}$  is in  $N(C)$  for a constant number of cells  $C \in \mathcal{C}$ . In addition to  $\mathcal{C}$ , Wang's algorithm [16] also computes the subsets  $P(C)$  and  $N(C)$  for all cells  $C \in \mathcal{C}$ , in a total of  $O(n \log n)$  time and  $O(n)$  space. Using the properties of  $\mathcal{C}$ , we can obtain the following lemma.

► **Lemma 2.** (1) Each arc of  $S$  is covered by the union of all cells of  $\mathcal{C}$ . (2) If an arc  $s \in S$  intersects  $C$ , then the center of  $s$  must be in one of the cells of  $N(C)$ . (3) Each arc of  $S$  only intersects a constant number of cells of  $\mathcal{C}$ . (4)  $\sum_{C \in \mathcal{C}} \sum_{C' \in N(C)} |P(C')| = O(n)$ .

**Proof.** To prove the first lemma statement, it suffices to show that for any point  $p$  of an arc of  $S$ ,  $p$  must be in a cell  $C$  of  $\mathcal{C}$ . Assume to the contrary this is not the case for a point  $p$  of an arc  $s \in S$ . Then, by property (3) of  $\mathcal{C}$ ,  $D_p \cap P = \emptyset$ . However, this is not true since  $D_p$  contains the center of  $s$ , which is in  $P$ . We thus obtain contradiction.

For the second lemma statement, consider a point  $p$  in a cell  $C$  and  $p$  is also on a segment  $s$ . Let  $q$  be the center of  $s$ . Since  $D_p$  contains  $q$ , by the property (4) of  $\mathcal{C}$ ,  $q$  must be in one of the cells of  $N(C)$ .

For the third lemma statement, consider an arc  $s$  whose center is in a cell  $C'$ . If  $s$  intersects a cell  $C$ , then by the second lemma statement,  $C'$  must be in  $N(C)$ . According to the property (5) of  $\mathcal{C}$ , each cell  $C'$  of  $\mathcal{C}$  is in  $N(C)$  for a constant number of cells  $C \in \mathcal{C}$ . This implies that  $s$  only intersects a constant number of cells of  $\mathcal{C}$ .

The fourth lemma statement simply follows the property (5) of  $\mathcal{C}$ . ◀

For each cell  $C \in \mathcal{C}$ , we use  $S(C)$  to denote the set of arcs of  $S$  whose centers are in  $P(C)$ .

With Lemma 2, we will compute the number of intersections of the arcs of  $S$  inside each cell  $C$  of  $\mathcal{C}$  and then add these numbers together. In what follows, we focus on one such cell  $C$ . Our goal is to compute the number of intersections of the arcs of  $S$  inside  $C$ . By Lemma 2, we only need to consider the arcs in  $S(C')$  for all  $C' \in N(C)$ . To this end, for each pair of cells  $(C_1, C_2)$  of  $N(C)$ , including the case where  $C_1 = C_2$ , we will compute the number of intersections inside  $C$  between the arcs of  $S(C_1)$  and the arcs of  $S(C_2)$ ; the sum of these numbers is the number of intersections inside  $C$  among all arcs of  $S$ . In the following, we discuss our algorithm for such a pair  $(C_1, C_2)$ .

We assume that  $C_1$  and  $C_2$  are two different cells since the other case can be solved by the same techniques. Let  $n_j = |S(C_j)|$ ,  $j = 1, 2$ . Since we are interested in the intersections inside  $C$ , for each arc  $s \in S(C_j)$ , we only keep its portions inside  $C$ . As the radius of  $s$  is 1

and  $C$  is an axis-parallel square of side length  $1/\sqrt{2}$ ,  $s$  has at most two (maximal) sub-arcs in  $C$ . Let  $S_j$  denote the set of these sub-arcs of  $S(C_j)$ . Thus,  $|S_j| \leq 2n_j$ . Our goal is to compute the number of intersections between the arcs of  $S_1$  and the arcs of  $S_2$ .

### 3.1 Counting intersections between $S_1$ and $S_2$ : Algorithm overview

To simplify the notation, we temporarily let  $n = |S_1| + |S_2|$  and  $S = S_1 \cup S_2$ . For each arc  $s \in S$ , by definition,  $s$  is inside  $C$ ; we extend both endpoints of  $s$  along the underlying circle of  $s$  until  $\partial C$ , and we refer to the new arc as the *extending arc* of  $s$ . Let  $S'$  denote the set of all extending arcs of  $S$ . Note that every two arcs of  $S'$  intersect at most twice.

We will start with computing a cutting of  $S'$  and then solve a *pseudo-trapezoid-restricted subproblem* in each cell of the cutting. Below we first describe our algorithm for solving the restricted subproblem. Consider a pseudo-trapezoid  $\tau$  of the cutting, which is in  $C$ . Let  $S(\tau)$  be the set of arcs of  $S$  intersecting  $\tau$  and let  $n_\tau = |S(\tau)|$ . Let  $S_1(\tau) = S_1 \cap S(\tau)$  and  $S_2(\tau) = S_2 \cap S(\tau)$ . The goal of the restricted subproblem is to compute the number of intersections between the arcs of  $S_1(\tau)$  and  $S_2(\tau)$  inside  $\tau$ . If we color the arcs of  $S_1(\tau)$  red and color those of  $S_2(\tau)$  blue, then we essentially have a bichromatic arc intersection problem: compute the number of intersections between red arcs and blue arcs.

We say that an arc  $s \in S(\tau)$  is a *short arc* if it has at least one endpoint in the interior of  $\tau$ . Otherwise,  $s$  is a *long arc*. Because the radius of  $s$  is 1,  $\tau$  is a pseudo-trapezoid inside  $C$  (and the radii of the circular arcs on the boundary of  $\tau$  are also 1), and  $C$  a square cell of side-length  $1/\sqrt{2}$ , the intersection of  $\tau$  and  $s$  consists of at most two (maximal) sub-arcs. Since we are only interested in intersections inside  $\tau$ , for each arc  $s \in S(\tau)$ , we trim it and only keep its portion (possibly two sub-arcs) inside  $\tau$ . If  $s$  has two trimmed sub-arcs in  $\tau$ , then both sub-arcs are treated as independent arcs, which are considered as long (resp., short) arcs if  $s$  is a long (resp., short) arc. A useful property of long arcs is that each long arc has both endpoints on  $\partial\tau$ . To simplify the notation, we still use  $S(\tau)$  to denote the set of all trimmed arcs in  $\tau$  and use  $n_\tau$  to denote the size of  $S(\tau)$ . Let  $m_\tau$  denote the number of short arcs of  $S(\tau)$ .

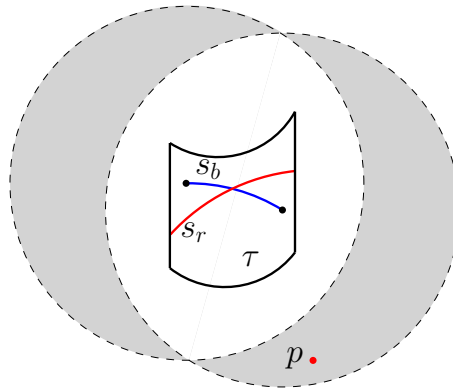
The arc intersections we are interested in can be classified into four types: (1) *long-red and long-blue intersections* (i.e., intersections between long red arcs and long blue arcs), (2) *short-red and short-blue intersections*, (3) *long-red and short-blue intersections*, and (4) *short-red and long-blue intersections*. For counting type (2) intersections, we use the algorithm of [3], which runs in  $O(m_\tau^{4/3+\epsilon})$  time. We will give an  $O(n_\tau \log^2 m_\tau + m_\tau \sqrt{n_\tau} \log m_\tau)$ -time algorithm for counting type (3) intersections in Section 3.2, and type (4) intersections can be counted in a similar way. Counting type (1) intersections is discussed in Section 3.3.

### 3.2 Counting type (3) intersections

We describe an algorithm to compute the number of type (3) intersections in  $O(n_\tau \log^2 m_\tau + m_\tau \sqrt{n_\tau} \log m_\tau)$  time. To this end, we further partition type (3) intersections into two sub-types: (3.1) a pair of a long red arc and a short blue arc that intersect only once; (3.2) a pair of a long red arc and a short blue arc that intersect twice. We first describe an algorithm to count type (3.1) intersections in  $O(m_\tau^2 / \log m_\tau + n_\tau \log^2 m_\tau)$  time.

#### 3.2.1 Counting type (3.1) intersections

For any long arc  $s \in S(\tau)$ , the intersection of its underlying circle  $\alpha(s)$  and  $\tau$  consists of at most two arcs (one of which is  $s$ ). If the intersection is only one arc, which is  $s$ , then we call  $s$  a *full arc*; otherwise it is a *partial arc*. We further partition type (3.1) intersections into



■ **Figure 3** The grey area is  $\text{lune}(s_b)$ .  $p$  is the center of  $s_r$ . The two dashed circles are unit circles centered at the two endpoints of  $s_b$ , respectively.

two sub-types: (3.1.1) a pair of a long red full arc and a short blue arc that intersect only once; (3.1.2) a pair of a long red partial arc and a short blue arc that intersect only once. Below, we first describe an algorithm to compute the number of type (3.1.1) intersections. The algorithm for type (3.1.2) is more complicated.

**Counting type (3.1.1) intersections.** Consider a short blue arc  $s_b$ . Let  $D_1$  and  $D_2$  be the two unit disks centered at the two endpoints of  $s_b$ , respectively. We call the region of  $D_1$  that is not in  $D_2$  a *lune*; similarly, region of  $D_2$  that is not in  $D_1$  is also a *lune*. We use  $\text{lune}(s_b)$  refer to the union of the two lunes (for simplicity, we consider  $\text{lune}(s_b)$  as a single lune defined by  $s_b$ ), i.e.,  $\text{lune}(s_b) = D_1 \cup D_2 - D_1 \cap D_2$ , the symmetric difference of  $D_1$  and  $D_2$  (e.g., the grey area in Fig. 3). By definition, a point is in  $\text{lune}(s_b)$  if and only if the unit disk centered at the point contains exactly one endpoint of  $s_b$ , implying that the unit circle centered at the point intersects  $s_b$  exactly once. Observation 3 is crucial to our algorithm for counting type (3.1.1) intersections in Lemma 4.

► **Observation 3.** For a short blue arc  $s_b$  and a long red full arc  $s_r$ ,  $s_r$  intersects  $s_b$  exactly once if and only if the center of  $s_r$  lies in  $\text{lune}(s_b)$ ; e.g., see Fig. 3.

**Proof.** Let  $p$  be the center of  $s_r$  and  $D$  be the underlying disk of  $s_r$ .

If  $s_r$  intersects  $s_b$  exactly once, then one endpoint of  $s_b$  must be inside  $D$  while the other is outside  $D$ . Therefore,  $p$  must be in  $\text{lune}(s_b)$ . Below we prove the other direction of the observation.

If  $p$  is in  $\text{lune}(s_b)$ , then by the definition of  $\text{lune}(s_b)$ , one endpoint of  $s_b$  is in  $D$  while the other is not. Hence,  $s_b$  must cross  $\partial D$  at exactly one point, say,  $q$ , which is the only intersection between  $s_b$  and  $\partial D$ . As  $s_b$  is in the cell  $\tau$ ,  $q \in \tau$ . Since  $q \in \partial D$ , we obtain that  $q \in \partial D \cap \tau$ . Further, since  $s_r$  is a long full arc, we have  $s_r = \partial D \cap \tau$ . Hence,  $q \in s_r$ . This implies that  $q \in s_r \cap s_b$ . Since  $s_r \in \partial D$  and  $s_b$  and  $\partial D$  has exactly one intersection, we conclude that  $s_b$  and  $s_r$  has exactly one intersection. ◀

► **Lemma 4.** Counting type (3.1.1) intersections takes  $O(m_\tau^2 / \log m_\tau + n_\tau \log m_\tau)$  time.

**Proof.** For notational convenience, let  $n = n_\tau$  and  $m = m_\tau$  in this proof. Recall that centers of red arcs are in the cell  $C_r \in \mathcal{C}$ . For each short blue arc  $s_b$ , since the radius of  $s_b$  is 1 and  $C_r$  is a square cell of the side-length  $1/\sqrt{2}$ , the boundary of the intersection of  $C_r$  and  $\text{lune}(s_b)$  has at most four circular arcs (which are on the unit circles centered at the two endpoints of  $s_b$ , respectively); let  $H$  be the set of these circular arcs defined by all short blue

arcs. Hence,  $|H| = O(m)$ . By Observation 3, it suffices to compute the number of lunes  $\text{lune}(s_b)$  containing the center of each long red arc. Let  $L$  denote the set of the lunes  $\text{lune}(s_b)$  of all short blue arcs  $s_b$ .

We build a hierarchical  $(1/r)$ -cutting  $\Xi_0, \Xi_1, \dots, \Xi_k$  on the arcs of  $H$ , with  $r = m/\log m$ . This can be done in  $O(mr)$  time by Theorem 1. For each cell  $\sigma \in \Xi_i$ ,  $1 \leq i \leq k$ , we define  $m_\sigma$  as the number of lunes of  $L$  that contain  $\sigma$  but do not contain the parent cell of  $\sigma$  in  $\Xi_{i-1}$  (i.e., the boundary of the lune crosses the parent cell). We can compute  $m_\sigma$  for all cells  $\sigma \in \Xi_i$ , for all  $i = 1, 2, \dots, k$ , as follows.

Recall that the algorithm of Theorem 1 for computing the hierarchical cutting also produces the sets  $H_\sigma$  for all cells  $\sigma \in \Xi_i$ ,  $0 \leq i \leq k$ , where  $H_\sigma$  is the subset of arcs of  $H$  crossing  $\sigma$ , in  $O(mr)$  time. As discussed in Section 2, the total size of these sets for all cells is  $O(mr)$ .

Next, we show that the values  $m_\sigma$  for all cells  $\sigma \in \Xi_i$ ,  $1 \leq i \leq k$  can be computed in  $O(mr)$  time using the sets  $H_\sigma$ . Define  $L(\sigma)$  as the subset of lunes of  $L$  that contain  $\sigma$  but does not contain the parent cell of  $\sigma$ . Hence,  $m_\sigma = |L(\sigma)|$ . Observe that one of the bounding arcs of each lune of  $L(\sigma)$  must cross  $\sigma'$ , where  $\sigma'$  is the parent of  $\sigma$  in  $\Xi_{i-1}$ . This implies that we can compute  $L(\sigma)$  and thus  $m_\sigma$  using  $H_{\sigma'}$  as follows. For each arc  $s \in H_{\sigma'}$ , we check whether the lune of  $L$  bounded by  $s$  contains  $\sigma$ ; if yes, we add the lune to  $L(\sigma)$ . Since each cell of the hierarchical cutting has  $O(1)$  child cells, the above algorithm for computing  $m_\sigma$  for all cells  $\sigma \in \Xi_i$ ,  $1 \leq i \leq k$ , runs in time linear in the total size of  $H_\sigma$  for all cells  $\sigma \in \Xi_i$ ,  $0 \leq i \leq k$ , which is  $O(mr)$  as explained above. As such, after the hierarchical cutting is computed,  $m_\sigma$  for all cells  $\sigma \in \Xi_i$ ,  $1 \leq i \leq k$ , can be computed in  $O(mr)$  time.

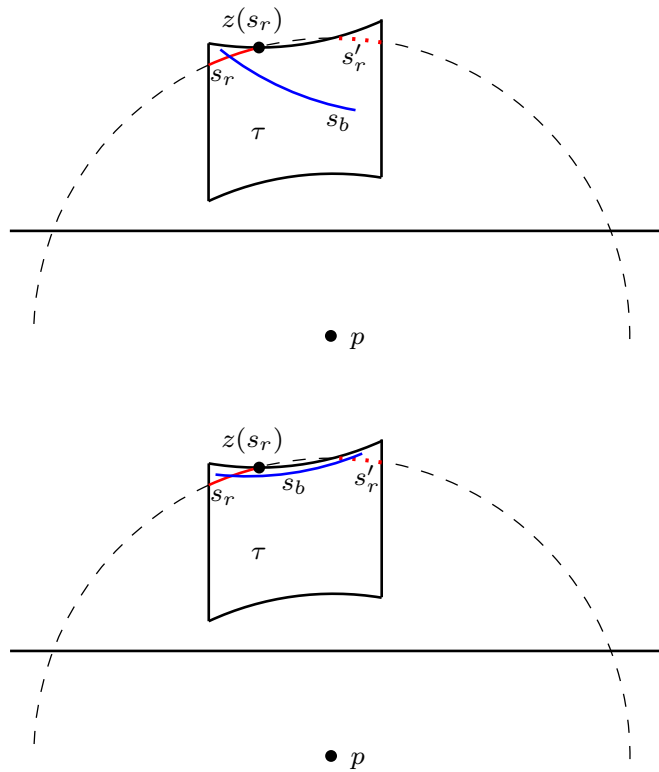
For each long red arc  $s$ , we locate the cell  $\sigma_i$  of  $\Xi_i$  containing  $q$  for each  $1 \leq i \leq k$ , where  $q$  is the center of  $s$ ; we add all these values  $m_{\sigma_i}$  to a total count  $m_s$  (initially,  $m_s = 0$ ). Finally, for the cell  $\sigma \in \Xi_k$ , for each arc  $s' \in H_\sigma$ , we check whether the lune of  $L$  bounded by  $s'$  contains  $q$ ; if yes, we increases  $m_s$  by one. It is not difficult to see the value  $m_s$  thus obtained is equal to the number of lunes of  $L$  containing  $q$ . Since  $|H_\sigma| = O(m/r) = O(\log m)$  for any  $\sigma \in \Xi_k$  and  $k = O(\log r)$ , the time of the above algorithm for computing  $m_s$  is  $O(\log m)$ . We apply the above algorithm for each long red arc, after which the number of type (3.1.1) intersections is computed. As there are at most  $n$  long red arcs, the total time of the algorithm is  $O(mr + n \log m)$ , which is  $O(m^2/\log m + n \log m)$  as  $r = m/\log m$ . ◀

**Counting type (3.1.2) intersections.** We now discuss type (3.1.2) intersections. Consider a long red partial arc  $s_r$ . Recall that the center of  $s_r$  is in the square cell  $C_r$ . By property (2) of  $\mathcal{C}$ , the two cells  $C_r$  and  $C$  are separated by an axis-parallel line. Depending on whether the line is horizontal or vertical, without loss of generality, we assume that  $C_r$  is either below  $C$  or to the right of  $C$ . Recall that the pseudo-trapezoid  $\tau$  is in  $C$ , and typically the upper (resp., lower) side of  $\tau$  is a unit circular arc while both the left and right sides of  $\tau$  are vertical segments. Since  $s_r$  is a partial arc, the underlying circle  $\alpha(s_r)$  intersects  $C$  at another arc, denoted by  $s'_r$ ; we call  $s'_r$  the *coupled arc* of  $s_r$  (and  $s_r$  is also the coupled arc of  $s'_r$ ).

Consider a type (3.1.2) intersection between a long red partial arc  $s_r$  and a short blue arc  $s_b$ . We further partition type (3.1.2) intersections into two types: (3.1.2.1)  $s_b$  does not intersect the coupled arc  $s'_r$  of  $s_r$ ; (3.1.2.2)  $s_b$  intersects  $s'_r$  (since  $s_r$  and  $s'_r$  are on the same circle and  $s_b$  intersects  $s_r$ ,  $s_b$  intersects  $s'_r$  exactly once in this case). We will present algorithms to count type (3.1.2.1) and type (3.1.2.2) intersections separately.

For each short-blue arc  $s_b$ , we use  $\text{lune}'(s_b)$  to denote the region of the plane outside the union of the two unit disks centered at the two endpoints of  $s_b$ .

Recall that  $C_r$  is either below  $C$  or to the right of  $C$ . Below we give observations for both cases, which lead to algorithms for counting type (3.1.2.1) and type (3.1.2.2) intersections.



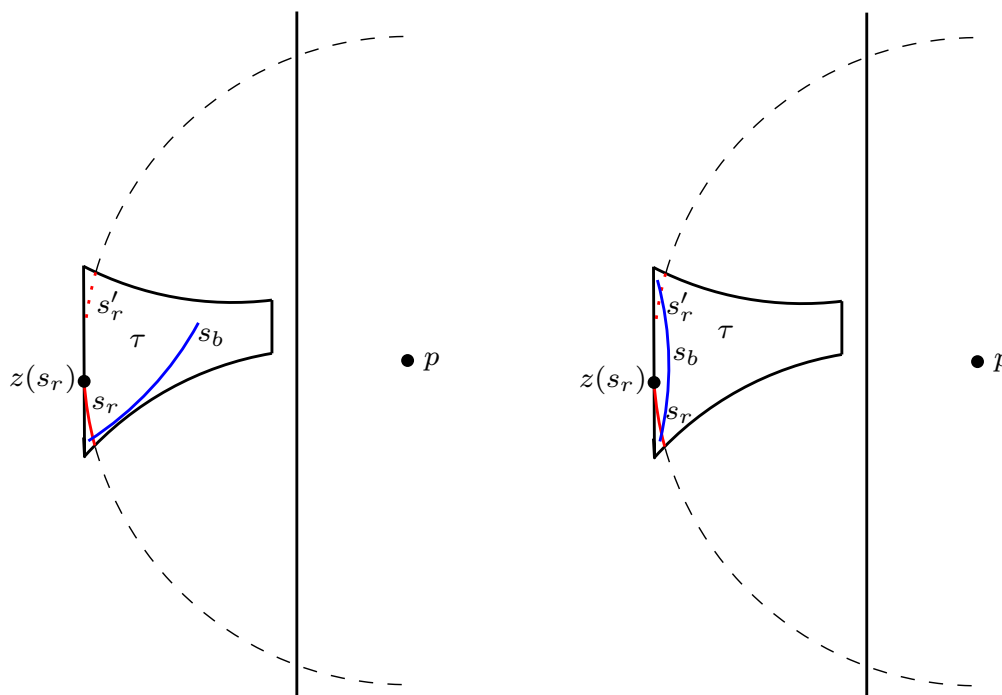
■ **Figure 4** Illustrating  $s_r$ ,  $s'_r$ , and  $z(s_r)$  for the case where  $C_r$  is below  $C$ . The point  $p$  is the center of  $s_r$  and the dashed curve is the upper semicircle of  $\alpha(s_r)$ . Top:  $s_r$  and  $s_b$  form a type (3.1.2.1) intersection (i.e.,  $s_r$  and  $s_b$  intersect exactly once and  $s'_r$  does not intersect  $s_b$ ). Bottom:  $s_r$  and  $s_b$  form a type (3.1.2.2) intersection (i.e.,  $s_b$  and  $s_r$  intersect exactly once and  $s_b$  intersects  $s'_r$ ).

Suppose  $C_r$  is below  $C$ . Then,  $s_r$  and  $s'_r$  are both on the upper semicircle of  $\alpha(s_r)$  (e.g., see Fig. 4). Thus, both  $s_r$  and  $s'_r$  are  $x$ -monotone, and  $s_r$  is either to the left or to the right of  $s'_r$ . Let  $z(s_r)$  denote the endpoint of  $s_r$  closer to  $s'_r$ , e.g.,  $z(s_r)$  is the right (resp., left) endpoint of  $s_r$  if  $s_r$  is to the left (resp., right) of  $s'_r$ . We have the following observation for type (3.1.2.1) and type (3.1.2.2) intersections. See the full paper for the proof.

► **Observation 5.** Let  $s_r$  be a long red partial arc and  $s_b$  a short blue arc. Suppose  $C_r$  is below  $C$ .

1.  $s_r$  and  $s_b$  form a type (3.1.2.1) intersection (i.e.,  $s_r$  and  $s_b$  intersect exactly once and  $s'_r$  does not intersect  $s_b$ ) if and only if the following two conditions hold: (1) the center of  $s_r$  lies in  $\text{lune}(s_b)$ ; (2) the endpoint of  $s_b$  not in the disk  $D(s_r)$  is to the left (resp., right) of  $z(s_r)$  if  $s_r$  is to the left (resp., right) of  $s'_r$ . See Fig. 4 (top).
2.  $s_r$  and  $s_b$  form a type (3.1.2.2) intersection (i.e.,  $s_b$  and  $s_r$  intersect exactly once and  $s_b$  intersects  $s'_r$ ) if and only if the following two conditions hold: (1) the center of  $s_r$  lies in  $\text{lune}'(s_b)$ ; (2) one endpoint of  $s_b$  is left of  $z(s_r)$  while the other endpoint is right of  $z(s_r)$ . See Fig. 4 (bottom).

If  $C_r$  is to the right of  $C$ ,  $s_r$  and  $s'_r$  are both in the left semicircle of  $\alpha(s_r)$ ; e.g., see Fig. 5. Thus, both  $s_r$  and  $s'_r$  are  $y$ -monotone, and  $s_r$  is either above or below  $s'_r$ . Let  $z(s_r)$  denote the endpoint of  $s_r$  closer to  $s'_r$ , e.g.,  $z(s_r)$  is upper (resp., lower) endpoint of  $s_r$  if  $s_r$  is below (resp., above)  $s'_r$ . We have the following observation, which is similar in spirit to Observation 5.



■ **Figure 5** Illustrating  $s_r$ ,  $s_r'$ , and  $z(s_r)$  where  $C_r$  is right of  $C$ . The point  $p$  is the center of  $s_r$  and the dashed curve is the left semicircle of  $\alpha(s_r)$ . Left:  $s_r$  and  $s_b$  form a type (3.1.2.1) intersection (i.e.,  $s_r$  and  $s_b$  intersect exactly once and  $s_r'$  does not intersect  $s_b$ ). Right:  $s_r$  and  $s_b$  form a type (3.1.2.2) intersection (i.e.,  $s_b$  and  $s_r$  intersect exactly once and  $s_b$  intersects  $s_r'$ ).

► **Observation 6.** Let  $s_r$  be a long-red partial arc and  $s_b$  a short-blue arc. Suppose  $C_r$  is right of  $C$ .

1.  $s_r$  and  $s_b$  form a type (3.1.2.1) intersection if and only if the following two conditions hold: (1) the center of  $s_r$  lies in  $\text{lune}(s_b)$ ; (2) the endpoint of  $s_b$  not in the disk  $D(s_r)$  is below (resp., above)  $z(s_r)$  if  $s_r$  is below (resp., above)  $s_r'$ . See Fig. 5 (left).
2.  $s_r$  and  $s_b$  form a type (3.1.2.2) intersection if and only if the following two conditions hold: (1) the center of  $s_r$  lies in  $\text{lune}'(s_b)$ ; (2) one endpoint of  $s_b$  is above  $z(s_r)$  while the other endpoint is below  $z(s_r)$ . See Fig. 5 (right).

Based on Observations 5 and 6, Lemmas 7 and 8 below compute the numbers of type (3.1.2.1) and type (3.1.2.2) intersections, respectively. Hence, the number of type (3.1.2) intersections can be computed in  $O(m_\tau^2/\log m_\tau + n_\tau \log^2 m_\tau)$  time.

► **Lemma 7.** Counting type (3.1.2.1) intersections takes  $O(m_\tau^2/\log m_\tau + n_\tau \log^2 m_\tau)$  time.

**Proof.** We assume that  $C_r$  is below  $C$  and the other case where  $C_r$  is right of  $C$  can be handled similarly. Our algorithm will be based on Observation 5(1). The algorithm is similar to Lemma 4 but with an additional level data structure. In what follows, we follow the notation in the proof of Lemma 4 and only point out the differences. For notational convenience, we let  $n = n_\tau$  and  $m = m_\tau$  in this proof.

We follow the same algorithm as in Lemma 4 but set  $r = m/\log^2 m$ . Recall that for each cell  $\sigma \in \Xi_i$ ,  $i = 1, 2, \dots, k$ , we have defined a subset  $L(\sigma) \subseteq L$  consisting of the lunes of  $L$  that contain  $\sigma$  but do not contain the parent cell of  $\sigma$  in  $\Xi_{i-1}$ . The total size of all these subsets is  $O(mr)$ . For each subset  $L(\sigma)$ , for each  $\text{lune}(s_b) \in L(\sigma)$ , since  $\text{lune}(s_b)$  contains  $\sigma$ , all unit

disks centered in  $\sigma$  contain exactly the same endpoint of  $s_b$  and let  $p(s_b, \sigma)$  denote the other endpoint; we use  $P(\sigma)$  to denote set of all these endpoints  $p(s_b, \sigma)$  for all  $\text{lune}(s_b) \in L(\sigma)$ . Note that  $|P(\sigma)| = |L(\sigma)|$ . We explicitly maintain  $P(\sigma)$ . Computing these subsets  $P(\sigma)$  for all cells can be done in  $O(mr)$  time by using the sets  $L(\sigma)$ . We further sort each set  $P(\sigma)$  by their  $x$ -coordinates. As the total size of  $P(\sigma)$  for all cells  $\sigma$  is  $O(mr)$ , the sorting for  $P(\sigma)$  of all cells  $\sigma$  can be done in  $O(mr \log m)$  time.

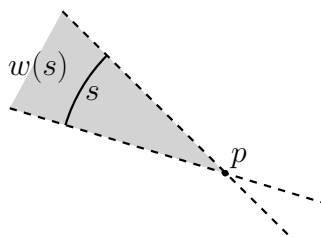
For each long red arc  $s_r$ , we first check whether it is a partial arc (this can be done in  $O(1)$  time). If not, we ignore it. If yes, we proceed as follows. Without loss of generality, we assume that  $s_r$  is to the left of  $s'_r$  and thus  $z(s_r)$  is the right endpoint of  $s_r$ . Let  $q$  be the center of  $s_r$ . We locate the cell  $\sigma_i$  of  $\Xi_i$  containing  $q$  for each  $1 \leq i \leq k$ . For each  $\sigma_i$ , according to Observation 5(1), we perform binary search on the sorted list of  $P(\sigma)$  to find the number of points of  $P(\sigma)$  to the left of  $z(s_r)$ , and let  $m_{\sigma_i}$  denote this number. We add all these values  $m_{\sigma_i}$  to a total count  $m_{s_r}$  (initially,  $m_{s_r} = 0$ ). Finally, for the cell  $\sigma \in \Xi_k$ , for each arc  $s' \in H_\sigma$ , we do the following. Note that  $s'$  is an arc of  $\text{lune}(s_b)$  for a short blue arc  $s_b$ . We check in  $O(1)$  time whether  $s_r$  and  $s_b$  form a type (3.1.2.1) intersection; if yes, we increase  $m_{s_r}$  by one. The value  $m_{s_r}$  thus obtained is equal to the number of type (3.1.2.1) intersections involving  $s_r$ . The time for computing  $m_{s_r}$  is  $O(\log^2 m)$ . We apply the above algorithm for each long red arc, after which the number of type (3.1.2.1) intersections is computed. The total runtime is  $O(mr \log m + n \log^2 m)$ , which is  $O(m^2 / \log m + n \log^2 m)$  as  $r = m / \log^2 m$ . ◀

► **Lemma 8.** *Counting type (3.1.2.2) intersections takes  $O(m_\tau^2 / \log m_\tau + n_\tau \log^2 m_\tau)$  time.*

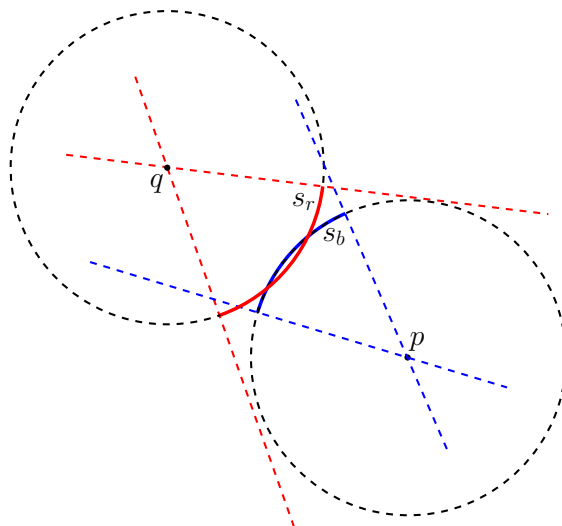
**Proof.** We assume that  $C_r$  is below  $C$  and the other case where  $C_r$  is right of  $C$  can be handled similarly. Our algorithm will be based on Observation 5(2). For notational convenience, let  $n = n_\tau$  and  $m = m_\tau$  in this proof.

We still follow the algorithm in the Lemma 4 but set  $r = m / \log^2 m$  and use  $\text{lune}'(s_b)$  to replace  $\text{lune}(s_b)$ . Note that the intersection of  $C_r$  and  $\text{lune}'(s_r)$  is still bounded by at most two circular arcs (other than the boundary of  $C_r$ ). Let  $H$  be the set of all these circular arcs for all short blue arcs. Let  $L$  denote the set of  $\text{lune}'(s_b)$  of all short blue arcs  $s_b$ . Next, we build a hierarchical cutting for  $H$  in the same way as in the proof of Lemma 4. We define  $L(\sigma)$  in the same way. Let  $B(\sigma)$  denote the set of short blue arcs  $s_b$  with  $\text{lune}'(s_b) \in L(\sigma)$ . Using  $L(\sigma)$ , we can obtain  $B(\sigma)$  for all cells  $\sigma \in \Xi_i$ ,  $i = 1, 2, \dots, k$ , in  $O(mr)$  time. For each arc  $s_b \in B(\sigma)$ , it defines an interval  $[x_1(s_b), x_2(s_b)]$ , where  $x_1(s_b)$  and  $x_2(s_b)$  are the  $x$ -coordinates of the left and right endpoints of  $s_b$ , respectively; let  $I(\sigma)$  denote the set of all these intervals for all arcs of  $B(\sigma)$ . We build an interval tree (Section 10.1 [5]) on the intervals of  $I(\sigma)$  in  $O(|I(\sigma)| \log |I(\sigma)|)$  time so that given a query  $x$ -coordinate  $x$ , we find in  $O(\log |I(\sigma)|)$  time the number of intervals of  $I(\sigma)$  containing  $x$ . As  $|I(\sigma)| = O(m)$  and the total size of the sets  $I(\sigma)$ 's of all cells in the cutting is  $O(mr)$ , the total time for building the interval trees is  $O(mr \log m)$ .

For each long red arc  $s_r$ , we first check whether it is a partial arc (this can be done in  $O(1)$  time). If not, we ignore it. If yes, we proceed as follows. Without loss of generality, we assume that  $s_r$  is to the left of  $s'_r$  and thus  $z(s_r)$  is the right endpoint of  $s_r$ . Let  $q$  be the center of  $s_r$ . We locate the cell  $\sigma_i$  of  $\Xi_i$  containing  $q$  for each  $i = 1, 2, \dots, k$ . For each  $\sigma_i$ , we query the interval tree of  $I(\sigma_i)$  to find number of intervals of  $I(\sigma_i)$  containing  $z(s_r)$ ; let  $m_{\sigma_i}$  denote this number. We add all these values  $m_{\sigma_i}$  to a total count  $m_{s_r}$  (initially,  $m_{s_r} = 0$ ). Finally, for the cell  $\sigma \in \Xi_k$ , for each arc  $s' \in H_\sigma$ , we do the following. Let  $s_b$  be the short-blue arc such that  $s'$  is an arc of  $\text{lune}'(s_b)$ . We check whether  $s_r$  and  $s_b$  form a type (3.1.2.1) intersection, which can be done in  $O(1)$  time; if yes, we increases  $m_{s_r}$  by



■ **Figure 6** Illustrating the wedge  $w(s)$  (the grey area):  $p$  is the center of  $s$ .



■ **Figure 7** Illustrating Observation 9:  $p$  is the center of  $s_b$  and  $q$  is the center of  $s_r$ .

one. The value  $m_{s_r}$  thus obtained is equal to the number of type (3.1.2.2) intersections involving  $s_r$ . The time for computing  $m_{s_r}$  is  $O(\log^2 m)$ . We apply the above algorithm for each long red arc, after which the number of type (3.1.2.2) intersections is computed. The total time of the algorithm is  $O(mr \log m + n \log^2 m)$ , which is  $O(m^2 / \log m + n \log^2 m)$  as  $r = m / \log^2 m$ . ◀

### 3.2.2 Counting type (3.2) intersections

We now compute the number of type (3.2) intersections. The runtime of the algorithm is  $O(m_\tau^2 \log^2 m_\tau + n_\tau \log^4 m_\tau)$ . Our goal is to compute the number of pairs of a long red arc  $s_r$  and a short blue arc  $s_b$  such that  $s_r$  and  $s_b$  intersect twice.

Let  $s$  be either a blue short arc or a long red arc. Since the radius of  $s$  is 1,  $s$  is contained in  $\tau$ , which is in  $C$ , and  $C$  is a square cell of side-length  $1/\sqrt{2}$ ,  $s$  does not span more than a semicircle of the underlying circle  $\alpha(s)$  of  $s$ . The two lines through the center of  $s$  and its two endpoints partition the plane into four wedges, one of which contains  $s$  completely (we use  $w(s)$  to denote the wedge; e.g., see Fig. 6). Observation 9 is crucial to our algorithm in Lemma 10 for counting type (3.2) intersections. See the full paper for the proof.

► **Observation 9.** Consider a short blue arc  $s_b$  and a long red arc  $s_r$ . If  $s_b$  and  $s_r$  intersect twice, then the following four conditions must hold (e.g., see Fig. 7): (1) the center of  $s_r$  is in the wedge  $w(s_b)$ ; (2) the center of  $s_b$  is in the wedge  $w(s_r)$ ; (3) the underlying disk  $D(s_r)$  of  $s_r$  does not contain either endpoint of  $s_b$ ; (4) the underlying circles  $\alpha(s_b)$  and  $\alpha(s_r)$

intersect. On the other hand, if the above four conditions all hold, then exactly one of the following two cases must happen: (1)  $s_b$  and  $s_r$  intersect twice; (2)  $s_r$  is a partial arc and  $s_b$  intersects both  $s_r$  and its coupled arc  $s'_r$ .

Notice that in the second case of Observation 9  $s_r$  and  $s_b$  form a type (3.1.2.2) intersection. Thus, if  $k_1$  is the number of pairs of a long red arc  $s_r$  and a short blue arc  $s_b$  that intersect twice, then  $k_1 = k_2 - k_3$ , where  $k_2$  is the number of pairs  $(s_r, s_b)$  that satisfy the four conditions in Observation 9 and  $k_3$  is the number of type (3.1.2.2) intersections. We have already computed  $k_3$  in Lemma 8. It remains to compute  $k_2$ , which is done in Lemma 10.

► **Lemma 10.** *Counting type (3.2) intersections takes  $O(m_\tau^2 \log^2 m_\tau + n_\tau \log^4 m_\tau)$  time.*

**Proof.** We first apply Lemma 8 and compute  $k_3$  in  $O(m_\tau^2 / \log m_\tau + n_\tau \log^2 m_\tau)$  time. As discussed before, it suffices to compute  $k_2$ , i.e., the number of pairs  $(s_r, s_b)$  that satisfy the four conditions in Observation 9. We show below that  $k_2$  can be computed in  $O(m_\tau^2 \log^2 m_\tau + n_\tau \log^4 m_\tau)$  time. For notational convenience, we let  $n = n_\tau$  and  $m = m_\tau$  in this proof.

Let  $B$  denote the set of short blue arcs and let  $R$  be the set of long red arcs of  $S(\tau)$ . For each short blue arc  $s_b$ , define  $A(s_b)$  as the common intersection of the wedge  $w(s_b)$  and  $\text{lune}'(s_b)$ . Note that the underlying disk of a long-red arc  $s_r$  does not contain either endpoint of a short-blue arc  $s_b$  if and only if the center of  $s_r$  is in  $\text{lune}'(s_b)$ . Hence, the first two conditions of Observation 9 are satisfied if and only if the center of  $s_r$  is in  $A(s_b)$ .

Recall that centers of all red arcs are in the square cell  $C_r$ . Let  $A'(s_b) = A(s_b) \cap C_r$ . Note that other than those on  $\partial C_r$ ,  $\partial A'(s_b)$  consists of at most two line segments and at most two circular arcs; we use *general-arcs* to refer to these arcs and segments (a segment can be considered as a circular arc of infinite radius). Hence,  $\partial A'(s_b)$  has at most four general-arcs. We call  $A'(s_b)$  the *interesting region* of  $s_b$ . Let  $I$  denote the set of interesting regions of all arcs of  $B$ . Let  $E$  denote the set of general-arcs of all regions of  $I$ . Hence,  $|E| = O(m)$ .

We compute a hierarchical  $(1/r)$ -cutting  $\Xi_0, \Xi_1, \dots, \Xi_k$  on the general-arcs of  $E$ , with  $r = m$  and a constant ratio  $\rho$  as described in Section 2. This can be done in  $O(mr)$  time by Theorem 1. For each cell  $\sigma \in \Xi_i$ ,  $1 \leq i \leq k$ , we define a canonical pair  $(B_\sigma, R_\sigma)$ , with  $B_\sigma \subseteq B$  and  $R_\sigma \subseteq R$ , so that centers of all arcs of  $R_\sigma$  lie in the interesting region of each short blue arc of  $B_\sigma$ . Specifically,  $R_\sigma$  consists of all long red arcs whose centers are located in the cell  $\sigma$  and  $B_\sigma$  consists of the arcs of  $B$  whose interesting regions contain  $\sigma$  but do not contain the parent cell of  $\sigma$  in  $\Xi_{i-1}$ . The canonical pairs of all cells of all cuttings  $\Xi_i$ ,  $1 \leq i \leq k$ , can be computed in  $O(mr + n \log m)$  time, as follows.

For the center  $p$  of each long red arc  $s_r$ , we locate the cell  $\sigma$  of  $\Xi_i$  containing  $p$ , for all  $1 \leq i \leq k$ , and add  $s_r$  to  $R_\sigma$ , which can be done in  $O(\log r)$  time. As such, computing the canonical sets  $R_\sigma$  for all cells  $\sigma$  takes  $O(n \log m)$  time in total. For computing  $B_\sigma$ , we can use an algorithm similar to the one in Lemma 4 (i.e., replacing  $\text{lune}(s_b)$  by the interesting region  $A'(s_b)$ ). More specifically, for each child cell  $\sigma \in \Xi_i$  of a cell  $\sigma'$  in  $\Xi_{i-1}$ , for each general-arc of  $E_{\sigma'}$  (i.e., the set of general-arcs of  $E$  crossing  $\sigma'$ ), suppose it is a general-arc of  $A'(s_b)$  of a blue arc  $s_b$ . We check whether  $A'(s_b)$  contains  $\sigma$ ; if yes, we add  $s_b$  to  $B_\sigma$ . In this way, computing all canonical subsets  $B_\sigma$  takes time proportional to the total size of the subsets  $E_{\sigma'}$  for all cells  $\sigma'$  of all cuttings  $\Xi_i$ ,  $i = 0, 1, \dots, k-1$ , which is  $O(mr)$ .

By the definition of canonical subset pairs  $(B_\sigma, R_\sigma)$ , to compute  $k_2$ , it suffices to compute the number of pairs of arcs  $(s_b, s_r)$  such that  $s_b \in B_\sigma$ ,  $s_r \in R_\sigma$ ,  $\alpha(s_b)$  intersects  $\alpha(s_r)$ , and the center of  $s_b$  is in  $w(s_r)$ , for all cells  $\sigma \in \Xi_i$ ,  $i = 1, 2, \dots, k$ . Below we describe an algorithm of  $O(m_\sigma^2 / \log m + n_\sigma \log m)$  time for one such cell  $\sigma \in \Xi_i$ , where  $m_\sigma = |B_\sigma|$  and  $n_\sigma = |R_\sigma|$ .

Recall that all centers of red arcs are in the square cell  $C_r$ . Define  $\mathcal{D}$  as the set of disks of radius 2 centered at the centers of all blue arcs of  $B_\sigma$ . Let  $P$  be the set of all centers of the arcs of  $R_\sigma$ ; if  $p \in P$  is the center of an arc  $s_r \in R_\sigma$ , we use  $s(p)$  to refer to  $s_r$ . It is

not difficult to see that the problem is equivalent to computing the number of pairs  $(p, D)$ ,  $p \in P$ ,  $D \in \mathcal{D}$ , such that  $p$  is contained in  $D$  and the center of  $D$  is in  $w(s(p))$ . To solve the problem, we use an algorithm similar to that for Lemma 4, with an additional level of range searching data structure for handling the constraint that the center of  $D$  is in  $w(s(p))$ . We briefly discuss it below. Note that  $m_\sigma = |\mathcal{D}|$  and  $n_\sigma = |P|$ .

For each disk  $D \in \mathcal{D}$ , the intersection of  $\partial D$  and  $C_r$  consists of at most two circular arcs. Let  $H$  denote the set of circular arcs  $\partial D \cap C_r$  for all disks  $D \in \mathcal{D}$ . We build a hierarchical  $(1/r')$ -cutting  $\Xi'_0, \Xi'_1, \dots, \Xi'_{k'}$  for  $H$ , with  $r' = m_\sigma / \log m$  and a constant ratio  $\rho_1$ , in  $O(m_\sigma r')$  time [10]. For each cell  $\beta \in \Xi'_i$ , for all  $i = 1, 2, \dots, k'$ , define  $\mathcal{D}(\beta)$  as the set of disks of  $\mathcal{D}$  that contain  $\beta$  but do not contain the parent cell of  $\beta$  in  $\Xi'_{i-1}$ . We can compute  $\mathcal{D}(\beta)$  explicitly for all cells of in  $O(m_\sigma r')$  time. We further build a range searching data structure on the centers of  $\mathcal{D}(\beta)$  for answering the following queries: Given a query wedge, compute the number of disk centers in the wedge. For this, we use a data structure of Matoušek's [14] (i.e., Lemma 6.1 [14] by setting  $p = 2$ ), which takes  $O(|\mathcal{D}(\beta)|^2 / \log^{1-\delta} |\mathcal{D}(\beta)|)$  preprocessing time for any  $\delta > 0$  and can answer each query in  $O(\log^2 |\mathcal{D}(\beta)|)$  time. In addition, for each cell  $\beta \in \Xi'_{k'}$ , we explicitly store the subset  $H_\beta$  of arcs of  $H$  crossing  $\beta$ .

All above can be done in  $O(m_\sigma r')$  time except that for building the wedge range searching data structures. We now analyze the total time for constructing these wedge range searching data structures. For each cell  $\beta \in \Xi_i$ , it holds that  $|\mathcal{D}(\beta)| = O(m_\sigma / \rho_1^{i-1})$ . Since  $\Xi_i$  has  $O(\rho_1^{2i})$  cells, the total time for constructing the data structures is big-O of  $\sum_{i=1}^{k'} \rho_1^{2i} \cdot m_\sigma^2 / \rho_1^{2(i-1)} / \log^{1-\delta} (m_\sigma / \rho_1^{i-1})$ , which is bounded by  $O(m_\sigma^2 \log m)$  as  $\rho_1$  is a constant.

For each point  $p \in P$ , we locate the cell  $\beta_i$  of  $\Xi'_i$  containing  $p$  for each  $i = 1, 2, \dots, k'$ ; using the wedge range searching data structure on the centers of  $\mathcal{D}(\beta_i)$ , we find the number  $m_{\beta_i}$  of centers of disks of  $\mathcal{D}(\beta_i)$  contained in the wedge  $w(s(p))$ . We add all these values  $m_{\beta_i}$  to a total count  $m_p$  (initially,  $m_p = 0$ ). Finally, for the cell  $\beta_k \in \Xi'_k$ , for each arc  $s \in H_{\beta_k}$ , we check whether the underlying disk of  $s$  contains  $p$  and the center of  $s$  is in  $w(s)$ ; if yes, we increases  $m_p$  by one. The value  $m_p$  thus obtained is equal to the number of disks  $D$  of  $\mathcal{D}$  that contain  $p$  and whose centers are in  $w(s(p))$ . The time for computing  $m_p$  is  $O(\log r' \log^2 m)$  as  $|H_\beta| = O(\log m)$  for any cell  $\beta \in \Xi'_k$ . The sum  $\sum_{p \in P} m_p$  is equal to the number of pairs  $(p, D)$ ,  $p \in P$ ,  $D \in \mathcal{D}$ , such that  $p$  is contained in  $D$  and the center of  $D$  is in  $w(s(p))$ . As  $n_\sigma = |P|$ , the total time of the algorithm is  $O(m_\sigma r' + m_\sigma^2 \log m + n_\sigma \log r' \log^2 m)$ , which is  $O(m_\sigma^2 \log m + n_\sigma \log^3 m)$  for  $r' = m_\sigma / \log m$ .

The above shows that processing each canonical subset pair  $(B_\sigma, R_\sigma)$  for a cell  $\sigma \in \Xi_i$  takes  $O(m_\sigma^2 \log m + n_\sigma \log^3 m)$  time, with  $m_\sigma = |B_\sigma|$  and  $n_\sigma = |R_\sigma|$ . Processing all cells  $\sigma \in \Xi_i$  for all  $i = 1, 2, \dots, k$  will compute the number  $k_2$ . We now analyze the total time. For each  $1 \leq i \leq k$ ,  $\sum_{\sigma \in \Xi_i} |R_\sigma| = n$  and  $m_\sigma \leq |E_{\sigma'}|$ , where  $\sigma'$  is the parent cell of  $\sigma$  in  $\Xi_{i-1}$ . Since  $|E_{\sigma'}| = O(m / \rho^{i-1})$ , we have  $m_\sigma = O(m / \rho^{i-1})$ . Because  $\Xi_i$  has  $O(\rho^{2i})$  cells, the total time for processing all cells in  $\Xi_i$  is on the order of  $\rho^{2i} \cdot m_\sigma^2 \log m + n \log^3 m \leq \rho^{2i} \cdot m^2 / \rho^{2i-2} \cdot \log m + n \log^3 m$ , which is bounded by  $O(m^2 \log m + n \log^3 m)$  as  $\rho$  is a constant. Since  $k = O(\log r)$  and  $r = m$ , the total time for processing all cells  $\sigma \in \Xi_i$  for all  $i = 1, 2, \dots, k$  is  $O(m^2 \log^2 m + n \log^4 m)$ . ◀

### 3.2.3 Counting type (3) intersections: a final step

The above shows that the number of type (3) intersections can be computed in  $O(m_\tau^2 \log^2 m_\tau + n_\tau \log^4 m_\tau)$  time. Using the result, we finally show that an alternative algorithm can compute the number of type (3) intersections in  $O(n_\tau \log^4 m_\tau + m_\tau \sqrt{n_\tau} \log^3 m_\tau)$  time.

If  $m_\tau < \sqrt{n_\tau} \log m_\tau$ , then  $m_\tau^2 \log^2 m_\tau \leq n_\tau \log^4 m_\tau$  and thus the runtime of the above algorithm is  $O(n_\tau \log^4 m_\tau)$ . Otherwise, we partition the short blue arcs into  $t$  groups of size  $m_\tau / t$ , with  $t = \frac{m_\tau}{\sqrt{n_\tau} \log m_\tau}$ . For each group, we apply the above algorithm on the group

with all long-red arcs. The total time is  $O(t \cdot (m_\tau^2/t^2 \log^2(m_\tau/t) + n_\tau \log^4(m_\tau/t)))$ , which is  $O(m_\tau \sqrt{n_\tau} \log^3 m_\tau)$ . As such, the number of type (3) intersections can be computed in  $O(n_\tau \log^4 m_\tau + m_\tau \sqrt{n_\tau} \log^3 m_\tau)$  time.

### 3.3 Counting type (1) intersections

Before giving our algorithm for counting the type (1) intersections, we analyze the total time for counting all intersections of other types in our original problem. Recall that initially we build a hierarchical  $(1/r)$ -cutting  $\Xi_0, \Xi_1, \dots, \Xi_k$  on the arcs of  $S'$  with  $n = |S'|$ . For each cell  $\tau$  of  $\Xi_k$ , we have the pseudo-trapezoid-restricted subproblem of counting the intersections of arcs of  $S(\tau)$  inside  $\tau$ , with  $n_\tau = |S(\tau)|$  and  $m_\tau$  as the number of short arcs of  $S(\tau)$ . By the definition of short and long arcs, we have  $m_\tau \leq n_\tau = O(n/r)$  and  $\sum_{\tau \in \Xi_k} m_\tau = O(n)$ . By setting  $r = n^{1/3}/(\log n)^{2/3}$ , we obtain the following lemma.

► **Lemma 11.** *By setting  $r = n^{1/3}/(\log n)^{2/3}$ , the total time for computing the number of type (2), (3), and (4) intersections for all cells  $\tau \in \Xi_k$  is bounded by  $O(n^{4/3}(\log n)^{10/3})$ .*

**Proof.** To solve the pseudo-trapezoid-restricted subproblem on each cell  $\tau \in \Xi_k$ , we have shown above that counting type (2) intersections can be done in  $O(m_\tau^{4/3+\epsilon})$  time and counting types (3) and (4) intersections takes  $O(n_\tau \log^4 m_\tau + m_\tau \sqrt{n_\tau} \log^3 m_\tau)$  time. Therefore, the total time for counting type (2) intersections for all cells  $\tau \in \Xi_k$  is  $O(\sum_{\tau \in \Xi_k} m_\tau^{4/3+\epsilon})$ . Since  $\Xi_k$  has  $O(r^2)$  cells,  $\sum_{\tau \in \Xi_k} m_\tau = O(n)$ , and  $m_\tau = O(n/r)$ ,  $\sum_{\tau \in \Xi_k} m_\tau^{4/3+\epsilon}$  achieves the maximum when  $r$   $m_\tau$ 's have value  $n/r$ , i.e.,  $\sum_{\tau \in \Xi_k} m_\tau^{4/3+\epsilon} = O(r \cdot (n/r)^{4/3+\epsilon})$ , which is  $O(n^{4/3+\epsilon}/r^{1/3+\epsilon})$ . Hence, the total time for counting type (2) intersections is  $O(n^{4/3+\epsilon}/r^{1/3+\epsilon})$ .

The total time for counting types (3) and (4) intersections is  $\sum_{\tau \in \Xi_k} (n_\tau \log^4 m_\tau + m_\tau \sqrt{n_\tau} \log^3 m_\tau)$ . Since  $\Xi_k$  has  $O(r^2)$  cells and  $n_\tau = O(n/r)$ , we have  $\sum_{\tau \in \Xi_k} n_\tau \log^4 m_\tau = O(r^2 \cdot n/r \cdot \log^4 n)$ , which is  $O(nr \log^4 n)$ . Since  $\sum_{\tau \in \Xi_k} m_\tau = O(n)$ , we can now derive  $\sum_{\tau \in \Xi_k} m_\tau \sqrt{n_\tau} \log^3 m_\tau = O(n \sqrt{\frac{n}{r}} \log^3 n)$ .

As such, the total time for computing the number of intersections of types (2-4) is  $O(n^{4/3+\epsilon}/r^{1/3+\epsilon} + nr \log^4 n + n \sqrt{\frac{n}{r}} \log^3 n)$ . Setting  $r = n^{1/3}/(\log n)^{2/3}$  makes the time bounded by  $O(n^{4/3}(\log n)^{10/3})$ . ◀

We now discuss how to compute the number of type (1) intersections, i.e., intersections between long red arcs and long blue arcs. If we consider each individual cell of the cutting  $\Xi_k$  as above, it would be difficult to achieve a satisfying time bound because each long arc may intersect many cells. Instead, we will count these intersections using the cuttings  $\Xi_0, \Xi_1, \dots, \Xi_k$  in the hierarchy. For each cell  $\sigma \in \Xi_i$  for any  $0 \leq i \leq k$ , we define long and short arcs of  $\sigma$  in the same way as before (i.e., suppose an arc  $s \in S$  intersects the interior of  $\sigma$ ; then  $s$  is a *long arc of  $\sigma$*  if  $s$  does not have either endpoint in the interior of  $\sigma$  and is a *short arc of  $\sigma$*  otherwise). For a long arc  $s$  of  $\sigma$ , we say that  $s$  is a *long-long arc* if  $i > 0$  and  $s$  is also a long arc of  $\sigma'$ , where  $\sigma'$  is the parent cell of  $\sigma$  in  $\Xi_{i-1}$ , and  $s$  is a *short-long arc* otherwise. A critical observation is that for a long arc  $s$  of any cell  $\sigma \in \Xi_k$ ,  $s$  must be a short-long arc of exactly one ancestor cell of  $\sigma$  since any arc must be a short arc of the only cell of  $\Xi_0$ , which is the entire square cell  $C$ .

Recall that a type (1) intersection refers to an intersection in a cell  $\sigma \in \Xi_k$  between a long red arc of  $\sigma$  and a long blue arc of  $\sigma$ . Consider a type (1) intersection  $q$  in a cell  $\sigma \in \Xi_k$  between a long red arc  $s_r$  and a long blue arc  $s_b$ . Observe that  $\sigma$  has one and only one ancestor cell  $\sigma'$  such that both  $s_r$  and  $s_b$  are long arcs of  $\sigma'$  but at least one of them is a short-long arc of  $\sigma'$  (also note that since  $\sigma'$  is an ancestor cell of  $\sigma$ ,  $\sigma'$  contains  $\sigma$  and thus

contains  $q$ ). This means that in order to count type (1) intersections, we can count, for all cells  $\sigma \in \Xi_i$ ,  $i = 0, 1, \dots, k$ , the type (1) intersections in  $\sigma$  involving at least one short-long arc, and more precisely, we can count the following three types of intersections: (1.1) intersections between long-long red arcs of  $\sigma$  and short-long blue arcs of  $\sigma$ ; (1.2) intersections between long-long blue arcs of  $\sigma$  and short-long red arcs of  $\sigma$ ; (1.3) intersections between short-long blue arcs and short-long red arcs of  $\sigma$ .

Consider a cell  $\sigma \in \Xi_i$ , for any  $0 \leq i \leq k$ . Let  $S_r$  be the set of all long red arcs of  $\sigma$  and  $S_b$  the set of all long blue arcs of  $\sigma$ . Let  $S_r^1$  and  $S_r^2$  be the subsets of long-long and short-long arcs of  $\sigma$  in  $S_r$ , respectively. Let  $S_b^1$  and  $S_b^2$  be the subsets of long-long and short-long arcs of  $\sigma$  in  $S_b$ , respectively. Let  $n_\sigma = |S_r| + |S_b|$  and  $m_\sigma = |S_r^2| + |S_b^2|$ . As such, the above three types of intersections are: (1.1) intersections between arcs of  $S_r^1$  and  $S_b^2$ , (1.2) intersections between arcs of  $S_r^2$  and  $S_b^1$ , and (1.3) intersections between arcs of  $S_r^2$  and  $S_b^2$ .

To compute the number of type (1.1) intersections, one could apply the algorithm in Section 3.2. However, the total time of the overall algorithm for all cells of  $\Xi_i$ ,  $i = 0, 1, \dots, k$ , cannot be bounded by  $O(n^{4/3} \cdot \text{poly}(\log n))$  since now we need to consider the cells in all cuttings  $\Xi_i$ ,  $i = 0, 1, \dots, k$ , not just the last cutting  $\Xi_k$  as before. Instead, we present a new algorithm in Lemma 12 (see the full paper for the proof). Type (1.2) intersections can be handled similarly as it is symmetric to type (1.1). For type (1.3), we actually also apply Lemma 12 because the algorithm works for any two sets of long arcs of  $\sigma$ , and the running time is also bounded as stated in Lemma 12 due to  $m_\sigma \leq n_\sigma$ .

► **Lemma 12.** *There exists an algorithm of  $O(m_\sigma \log^4 n_\sigma + n_\sigma \log^5 n_\sigma + n_\sigma^{2/3} m_\sigma^{2/3} (\log n_\sigma)^{13/3})$  time to compute the number of type (1.1) intersections.*

Using Lemma 12, the following lemma analyzes the total time of our algorithm for computing the number of type (1) intersections.

► **Lemma 13.** *Counting type (1) intersections takes  $O(n^{4/3} \log^{16/3} n)$  time.*

**Proof.** As discussed before, the number of type (1.1) intersections can be computed once the subproblems stated in Lemma 12 are solved for all cells  $\sigma \in \Xi_i$ ,  $i = 0, 1, \dots, k$ . We next analyze the total time for solving these subproblems. We follow the notation as defined before. Note that  $n_\sigma = O(\frac{n}{\rho^i})$  and  $\Xi_i$  has  $O(\rho^{2i})$  cells.

We first claim that  $\sum_{\sigma \in \Xi_i} m_\sigma = O(n)$  for all  $0 \leq i \leq k$ . Indeed, it is obviously true that  $\sum_{\sigma \in \Xi_i} m_\sigma = O(n)$  for  $i = 0$ . We assume that  $i > 0$ . Consider a short-long arc  $s$  of  $\sigma$ . By definition,  $s$  is a short arc of the parent cell  $\sigma' \in \Xi_{i-1}$  of  $\sigma$ , implying that  $s$  has at least one endpoint in the interior of  $\sigma'$ . Let  $\sigma_1$  and  $\sigma_2$  be the two cells of  $\Xi_{i-1}$  that contain the two endpoints of  $s$ , respectively ( $\sigma_1 = \sigma_2$  is possible). According to the above discussion, if  $s$  is short-long arc of  $\sigma$  for some cell  $\sigma \in \Xi_i$ , then  $\sigma$  must be a child cell of either  $\sigma_1$  or  $\sigma_2$ . As  $\sigma_j$  has  $O(1)$  cells in  $\Xi_i$ , for  $j = 1, 2$ ,  $s$  can be a short-long arc of at most  $O(1)$  cells  $\sigma \in \Xi_i$ . As such,  $\sum_{\sigma \in \Xi_i} m_\sigma = O(n)$  holds.

We now consider the three terms in the time complexity of Lemma 12 separately.

1. For the first term,  $\sum_{i=0}^k \sum_{\sigma \in \Xi_i} m_\sigma \log^4 n_\sigma = O(\sum_{i=1}^k n \log^4 n) = O(n \log^5 n)$ , for  $k = O(\log r)$  and  $r = n^{1/3}/(\log n)^{2/3}$ .
2. For the second term, we have  $\sum_{i=0}^k \sum_{\sigma \in \Xi_i} n_\sigma \log^5 n_\sigma = O(\sum_{i=1}^k \rho^{2i} \cdot \frac{n}{\rho^i} \log^5 n)$ , which is bounded by  $O(nr \log^5 n)$  as  $r = \Theta(\rho^k)$ .
3. For the third term,  $\sum_{\sigma \in \Xi_i} n_\sigma^{2/3} m_\sigma^{2/3} (\log n_\sigma)^{13/3}$  is big-O of  $\sum_{\sigma \in \Xi_i} (\frac{n}{\rho^i})^{2/3} m_\sigma^{2/3} (\log n)^{13/3} = (\frac{n}{\rho^i})^{2/3} \cdot (\log n)^{13/3} \cdot \sum_{\sigma \in \Xi_i} m_\sigma^{2/3}$ . As  $\sum_{\sigma \in \Xi_i} m_\sigma = O(n)$  and  $\Xi_i$  has  $O(\rho^{2i})$  cells, by Hölder's Inequality,  $\sum_{\sigma \in \Xi_i} m_\sigma^{2/3} = O(\rho^{2i} \cdot (\frac{n}{\rho^{2i}})^{2/3})$ . As such, we can obtain that  $\sum_{\sigma \in \Xi_i} n_\sigma^{2/3} m_\sigma^{2/3} (\log n_\sigma)^{13/3}$  is big-O of

$(\frac{n}{\rho^i})^{2/3} \cdot (\log n)^{13/3} \cdot \rho^{2i} \cdot (\frac{n}{\rho^{2i}})^{2/3}$ , which is bounded by  $O(n^{4/3}(\log n)^{13/3})$  as  $\rho$  is a constant. Hence  $\sum_{i=0}^k \sum_{\sigma \in \Xi_i} n_\sigma^{2/3} m_\sigma^{2/3} (\log n_\sigma)^{13/3} = O(n^{4/3}(\log n)^{16/3})$ , for  $k = O(\log r)$  and  $r = n^{1/3}/(\log n)^{2/3}$ .

Therefore, the total time for solving the subproblems for all cells  $\sigma \in \Xi_i$  for all  $i = 0, 1, \dots, k$  is  $O(n \log^5 n + nr \log^5 n + n^{4/3} \log^{16/3} n)$ , which is  $O(n^{4/3} \log^{16/3} n)$  as  $r = n^{1/3}/(\log n)^{2/3}$ . Hence, the number of type (1.1) intersections can be computed in time bounded by  $O(n^{4/3} \log^{16/3} n)$ .

As discussed before, Lemma 12 is also applicable to counting type (1.2) and type (1.3) intersections with asymptotically the same time complexity. As such, the total number of type (1) intersections can be computed in  $O(n^{4/3} \log^{16/3} n)$  time. ◀

By Lemmas 11 and 13, we conclude that the total number of intersections all types (i.e., the number of intersections in the square cell  $C$ ) can be computed in  $O(n^{4/3} \log^{16/3} n)$  time. Note that the time complexities in the two lemmas imply that computing type (1) intersections is the bottleneck.

### 3.4 Putting it all together

The above solves the problem of computing the number of intersections in the square cell  $C$  between the arcs of  $S(C_1)$  and the arcs of  $S(C_2)$ . The time of the algorithm is  $O((n_1 + n_2)^{4/3} \log^{16/3}(n_1 + n_2))$ , where  $n_1 = |S(C_1)|$  and  $n_2 = |S(C_2)|$ . The algorithm is based on the assumption that  $C_1 \neq C_2$ . If  $C_1 = C_2$ , then the problem becomes computing the intersections in  $C$  among all arcs of  $S(C_1)$ . Our algorithm can be easily adapted to this case. For example, we can duplicate all arcs of  $S(C_1)$  and make the duplications as  $S(C_2)$ , and then apply the algorithm (one subtle issue that in this case an arc of  $S(C_1)$  and its duplication in  $S(C_2)$  should not be considered intersected).

Enumerating all pairs  $(C_1, C_2)$  of  $N(C)$  and solving the problem as above can compute the number of all arc intersections in  $C$ . Processing all cells  $C \in \mathcal{C}$  can finally compute the number of all arc intersections of  $S$  in the plane. As such, the total time of the overall algorithm is  $O(\sum_{C \in \mathcal{C}} \sum_{(C_1, C_2) \in N(C)} (n_1 + n_2)^{4/3} \log^{16/3}(n_1 + n_2))$ . Recall that  $|S(C')| = |P(C')|$  for any cell  $C' \in \mathcal{C}$ . Because  $|N(C)| = O(1)$ , we have  $\sum_{(C_1, C_2) \in N(C)} (n_1 + n_2)^{4/3} \log^{16/3}(n_1 + n_2) = O(n_C^{4/3} \log^{16/3} n)$ , where  $n_C = \sum_{C' \in N(C)} |P(C')|$ . By Lemma 2(4),  $\sum_{C \in \mathcal{C}} n_C = O(n)$ . Hence,  $\sum_{C \in \mathcal{C}} n_C^{4/3} = O(n^{4/3})$ . As such, the runtime of the overall algorithm is  $O(n^{4/3} \log^{16/3} n)$ .

► **Theorem 14.** *The number of intersections among a set of  $n$  circular arcs of the same radius in the plane can be computed in  $O(n^{4/3} \log^{16/3} n)$  time.*

We further improve the algorithm when the number of intersections of the arcs is small.

► **Theorem 15.** *The number of intersections among a set of  $n$  circular arcs of the same radius in the plane can be computed in  $O(n^{1+\epsilon} + K^{1/3} n^{2/3} (\frac{n^2}{n+K})^\epsilon \log^{16/3} n)$  time, for any  $\epsilon > 0$ , where  $K$  is the number of intersections of all arcs.*

**Proof.** Let  $S$  denote the set of all arcs. Let  $r = n^2/(n+K)$ . By Theorem 1, we first compute a  $(1/r)$ -cutting  $\Xi$  for  $S$  of size  $O((n^2/(n+K))^{1+\epsilon})$  in  $O(nr^\epsilon + Kr/n)$  time, so that each cell of  $\Xi$  is intersected by at most  $n/r = (n+K)/n$  arcs of  $S$ . Note that  $nr^\epsilon + Kr/n = O(n^{1+\epsilon})$ . Hence, constructing the cutting takes  $O(n^{1+\epsilon})$  time. For each cell  $\sigma$  of  $\Xi$ , we compute the number of intersections of arcs inside  $\sigma$ , in  $O(((n+K)/n)^{4/3} \log^{16/3} n)$  time by Theorem 14. The total time of the overall algorithm is therefore on the order of  $n^{1+\epsilon} + (\frac{n^2}{n+K})^{1+\epsilon} \cdot (\frac{n+K}{n})^{4/3} \cdot \log^{16/3} n$ , which is bounded by  $O(n^{1+\epsilon} + K^{1/3} n^{2/3} (\frac{n^2}{n+K})^\epsilon \log^{16/3} n)$ .

The above algorithm depends on the unknown value  $K$ . To overcome of the issue, we guess the value of  $K$  by the trick of doubling. We start with  $K' = K_0$  for a constant  $K_0$ , and run the algorithm with  $K'$ . If the algorithm takes too long, then our guess is too low and we double  $K'$ . Using this strategy, the algorithm is expected to stop within a constant number of rounds when  $K'$  is larger than  $K$  for the first time. Hence, the total time is asymptotically the same as if we had plugged in the right value of  $K$ , except that we call the cutting construction algorithm for  $O(\log K)$  (which is  $O(\log n)$ ) times. Hence, the total time the algorithm spends on constructing the cutting is still  $O(n^{1+\epsilon})$  as the  $\log n$  factor is absorbed by  $\epsilon$ . Therefore, the total time of the overall algorithm is still  $O(n^{1+\epsilon} + K^{1/3}n^{2/3}(\frac{n^2}{n+K})^\epsilon \log^{16/3} n)$ . ◀

#### 4 Concluding remarks

In this paper, we present an  $O(n^{4/3} \log^{16/3} n)$  time algorithm for computing the number of intersections among a set of  $n$  circular arcs of the same radius, which nearly matches the  $\Omega(n^{4/3})$  lower bound of Erickson in the partition algorithm model [11]. Our algorithm involves a partition of the problem into several cases. While this partition makes the algorithm somewhat tedious, it appears necessary to treat these different cases separately, which may be due to the nature of the problem. Indeed, this partition is one of the main reasons that our algorithm is superior to the previous work of Agarwal, Pellegrini, and Sharir [3]. It remains unclear whether this partition can be simplified.

Some of our techniques and observations may be useful for solving other related problems. We demonstrate two examples below.

**The bichromatic problem.** It is straightforward to adapt our algorithm to solving the bichromatic problem, i.e., given a set of blue circular arcs and another set of red circular arcs of the same radius, with  $n$  as the total number of all arcs, compute the number of intersections between red arcs and blue arcs. Indeed, our algorithm for solving the problem for a single cell  $C \in \mathcal{C}$  in Sections 3.1, 3.2, and 3.3 is for the bichromatic problem. More specifically, we first compute the set  $\mathcal{C}$  of square cells in a similar way as before. Then, for each cell  $C \in \mathcal{C}$ , for each pair of cells  $(C_1, C_2)$  of  $N(C)$ , we compute the number of intersections between the blue arcs of  $S(C_1)$  and the red arcs of  $S(C_2)$  as well as the number of intersections between the red arcs of  $S(C_1)$  and the blue arcs of  $S(C_2)$ , by applying our algorithm in Sections 3.1, 3.2, and 3.3. In this way, the bichromatic problem can be solved in  $O(n^{4/3} \log^{16/3} n)$  time. The time can then be further reduced to  $O(n^{1+\epsilon} + K^{1/3}n^{2/3}(\frac{n^2}{n+K})^\epsilon \log^{16/3} n)$  time, for any  $\epsilon > 0$ , where  $K$  is the number of intersections of all arcs.

Alternatively, one can solve the bichromatic problem by using our algorithm for the monochromatic case as a black-box. First, we compute the number of intersections of all arcs, denoted by  $K$ . Then, we compute the number of intersections among the red arcs, denoted by  $K_r$ , and also compute the number of intersections among the blue arcs, denoted by  $K_b$ . Observe that  $K - K_r - K_b$  is the number of bichromatic intersections. The total runtime of the algorithm is asymptotically the same as that for the monochromatic case.

**The segment case.** Suppose  $S$  is a set of  $n$  line segments in the plane. To compute the number of intersections of  $S$ , using Chan and Zheng's recent  $O(n^{4/3})$  time algorithm [8] and the techniques of Theorem 15, we can solve the problem in  $O(n^{1+\epsilon} + K^{1/3}n^{2/3}(\frac{n^2}{n+K})^\epsilon)$  time. Indeed, applying Theorem 1 with  $r = n^2/(n+K)$ , we first compute a  $(1/r)$ -cutting  $\Xi$  for  $S$  of size  $O((n^2/(n+K))^{1+\epsilon})$  in  $O(nr^\epsilon + Kr/n)$  time, so that each cell of  $\Xi$  is intersected by at most  $n/r = (n+K)/n$  segments of  $S$ . Since  $nr^\epsilon + Kr/n = O(n^{1+\epsilon})$ , constructing the cutting takes

$O(n^{1+\epsilon})$  time. For each cell  $\sigma$  of  $\Xi$ , we compute the number of intersections of segments inside  $\sigma$ , in  $O(((n+K)/n)^{4/3})$  time by Chan and Zheng’s algorithm [8]. Hence, the total time is on the order of  $n^{1+\epsilon} + (\frac{n^2}{n+K})^{1+\epsilon} \cdot (\frac{n+K}{n})^{4/3}$ , which is bounded by  $O(n^{1+\epsilon} + K^{1/3}n^{2/3}(\frac{n^2}{n+K})^\epsilon)$ . The above algorithm depends on the unknown value  $K$ . To overcome of the issue, we again use the doubling technique as in Theorem 15. As the analysis in Theorem 15, the total time of the algorithm is still  $O(n^{1+\epsilon} + K^{1/3}n^{2/3}(\frac{n^2}{n+K})^\epsilon)$ .

We also remark that by plugging Chan and Zheng’s algorithm [8] into a randomized algorithm of de Berg and Schwarzkopf [6] (i.e., replacing Chazelle’s algorithm [10] with Chan and Zheng’s algorithm [8] in Theorem 3 [6]), the number of the intersections of  $S$  can be computed in  $O(n \log n \log K + K^{1/3}n^{2/3})$  expected time. The algorithm is similar as above, except that it uses a cutting with slightly better bounds; specifically, de Berg and Schwarzkopf (Theorem 1 [6]) gave a randomized algorithm that can construct a  $(1/r)$ -cutting of size  $O(r + Kr^2/n^2)$  for  $S$  in  $O(n \log r + Kr/n)$  expected time.

---

## References

- 1 Pankaj K. Agarwal. Partitioning arrangements of lines II: Applications. *Discrete and Computational Geometry*, 5:533–573, 1990. doi:10.1007/BF02187809.
- 2 Pankaj K. Agarwal, Boris Aronov, Micha Sharir, and Subhash Suri. Selecting distances in the plane. *Algorithmica*, 9:495–514, 1993. doi:10.1007/BF01187037.
- 3 Pankaj K. Agarwal, Marco Pellegrini, and Micha Sharir. Counting circular arc intersections. *SIAM Journal on Computing*, 22:778–793, 1993. doi:10.1137/0222050.
- 4 Pankaj K. Agarwal and Micha Sharir. Pseudoline arrangements: Duality, algorithms, and applications. *SIAM Journal on Computing*, 34:526–552, 2005. doi:10.1137/S0097539703433900.
- 5 Mark de Berg, Otfried Cheong, Marc J. van Kreveld, and Mark H. Overmars. *Computational Geometry — Algorithms and Applications*. Springer-Verlag, Berlin, 3rd edition, 2008.
- 6 Mark de Berg and Otfried Schwarzkopf. Cuttings and applications. *International Journal of Computational Geometry and Applications*, 5:343–355, 1995. doi:10.1142/S0218195995000210.
- 7 Timothy M. Chan, Pingan Cheng, and Da Wei Zheng. Semialgebraic range stabbing, ray shooting, and intersection counting in the plane. In *Proceedings of the 40th International Symposium on Computational Geometry (SoCG)*, pages 33:1–33:15, 2024. doi:10.4230/LIPIcs.SoCG.2024.33.
- 8 Timothy M. Chan and Da Wei Zheng. Hopcroft’s problem, log-star shaving, 2D fractional cascading, and decision trees. *ACM Transactions on Algorithms*, 2023. doi:10.1145/3591357.
- 9 Bernard Chazelle. Reporting and counting segment intersections. *Journal of Computer and System Sciences*, 32:156–182, 1986. doi:10.1016/0022-0000(86)90025-5.
- 10 Bernard Chazelle. Cutting hyperplanes for divide-and-conquer. *Discrete and Computational Geometry*, 9(2):145–158, 1993. doi:10.1007/BF02189314.
- 11 Jeff Erickson. New lower bounds for Hopcroft’s problem. *Discrete and Computational Geometry*, 16:389–418, 1996. doi:10.1007/BF02712875.
- 12 Leonidas J. Guibas, Mark H. Overmars, and Micha Sharir. Counting and reporting intersections in arrangements of line segments. Technical Report 434, Department of Computer Science, New York University, 1989.
- 13 Matthew J. Katz and Micha Sharir. An expander-based approach to geometric optimization. *SIAM Journal on Computing*, 26(5):1384–1408, 1997. doi:10.1137/S0097539794268649.
- 14 Jiří Matoušek. Range searching with efficient hierarchical cuttings. *Discrete and Computational Geometry*, 10(1):157–182, 1993. doi:10.1007/BF02573972.
- 15 Marco Pellegrini. On counting pairs of intersecting segments and off-line triangle range searching. *Algorithmica*, 17:380–398, 1997. doi:10.1007/BF02523679.
- 16 Haitao Wang. Unit-disk range searching and applications. *Journal of Computational Geometry*, 14:343–394, 2023. doi:10.20382/jocg.v14i1a13.