The 2-Center Problem in Three Dimensions*

Pankaj K. Agarwal† Rinat Ben Avraham‡ Micha Sharir§

Abstract

Let P be a set of n points in $\mathbb{R}^3$. The *2-center problem* for P is to find two congruent balls of minimum radius whose union covers P. We present a randomized algorithm for computing a 2-center of P that runs in $O(\beta(r^*)n^2 \log^4 n \log\log n)$ expected time; here $\beta(r) = 1/(1 - r/r_0)^3$, r^* is the radius of the 2-center balls of P, and r_0 is the radius of the smallest enclosing ball of P. The algorithm is near quadratic as long as r^* is not too close to r_0, which is equivalent to the condition that the centers of the two covering balls be not too close to each other. This improves an earlier slightly super-cubic algorithm of Agarwal, Efrat, and Sharir [2] (at the cost of making the algorithm performance depend on the center separation of the covering balls).

*Work on this paper has been supported by Grant 2006/194 from the U.S.-Israeli Binational Science Foundation. Work by Pankaj Agarwal is also supported by NSF under grants CCF-06 -35000, IIS-07-13498, CCF-09-40671, and CCF-10-12254, by ARO grants W911NF-07-1-0376 and W911NF-08-1-0452, and by an ERDC contract W9132V-11-C-0003. Work by Micha Sharir has also been supported by NSF Grants CCF-05-14079 and CCF-08-30272, by Grants 155/05 and 338/09 from the Israel Science Fund, by the Israeli Centers for Research Excellence (I-CORE) program (center no. 4/11), and by the Hermann Minkowski–MINERVA Center for Geometry at Tel Aviv University.

A preliminary version of this paper appeared in *Proc. 26th Sympos. on Computational Geometry* 2010, pp. 87–96.

†Department of Computer Science, Box 90129, Duke University, Durham, NC 27708-0129, USA; pankaj@cs.duke.edu

‡School of Computer Science, Tel Aviv University, Tel Aviv 69978, Israel; rinatba@gmail.com

§School of Computer Science, Tel Aviv University, Tel Aviv 69978, Israel; and Courant Institute of Mathematical Sciences, New York University, New York, NY 10012, USA; michas@post.tau.ac.il

1 Introduction

Let $P = \{p_1, \ldots, p_n\}$ be a set of n points in $\mathbb{R}^3$. The *2-center problem* for P is to find two congruent balls of minimum radius whose union covers P. This is a special case of the general k-center problem in $\mathbb{R}^d$, which calls for covering a set P of n points in $\mathbb{R}^d$ by k congruent balls of minimum radius. If k is part of the input, the problem is known to be NP-complete [25] even for $d = 2$, so the complexity of algorithms for solving the k-center problem, for any fixed k, is expected to increase more than polynomially in k. Agarwal and Procopiuc showed that the k-center problem in $\mathbb{R}^d$ can be solved in $n^{O(k^{1-1/d})}$ time [3], improving upon a naive $n^{O(k)}$-solution. Several efficient approximation algorithms and heuristics have been proposed for the k-center problem; they run in near-linear time (in terms of n) [3, 5, 6, 19].

The intractability of the general k-center problem has also led to extensive work on developing efficient (exact) k-center algorithms for small values of k. The 1-center problem (also known as the *smallest enclosing ball* problem) is known to be an LP-Type problem [24], and can thus be solved in $O(n)$ randomized expected time in any fixed dimension, and also in deterministic linear time [11]. If d is not fixed, the 2-center problem in $\mathbb{R}^d$ is NP-Complete [26]. The 2-center problem in $\mathbb{R}^2$ has a relatively rich history, mostly in the past two decades. Hershberger and Suri [20] showed that the decision problem of determining whether P can be covered by two disks of a given radius r can be solved in $O(n^2 \log n)$ time. This has led to several near-quadratic algorithms [4, 21] that solve the optimization problem, the best of which is a deterministic algorithm by Jaromczyk and Kowaluk [21] that runs in $O(n^2 \log n)$ time. Sharir [28] considerably improved these bounds and obtained a deterministic algorithm with $O(n \log^9 n)$ running time. His algorithm combines several geometric techniques, including parametric searching, searching in monotone matrices, and dynamic maintenance of planar configurations. Eppstein [16] slightly improved this algorithm, using a randomized approach, resulting in a solution with $O(n \log^2 n)$ expected time. Later, Chan [9] made another improvement and gave a deterministic algorithm with $O(n \log^2 n \log^2 \log n)$ running time.

The only earlier work on the 2-center problem in $\mathbb{R}^3$ we are aware of is by Agarwal, Efrat, and Sharir [2], which presents an algorithm with $O(n^{3+\varepsilon})$ running time, for any $\varepsilon > 0$. It uses a rather complicated data structure for dynamically maintaining upper and lower envelopes of bivariate functions.

We present a randomized algorithm for the 2-center problem in $\mathbb{R}^3$ that runs in $O(\beta(r^*)n^2 \log^4 n \log\log n)$ expected time, where $\beta(r) = 1/(1 - r/r_0)^3$, r_0 is the radius of the 1-center ball of P, and $r^* \leq r_0$ is the common radius of the 2-center balls of P. The algorithm is based on some of the ideas in Sharir's planar algorithm [28] and their refinements [9, 16], but requires several new techniques. As in the previous algorithms for the 2-center problem, we first present an algorithm for the decision problem: given $r > 0$, determine whether P can be covered by two balls of radius r. We then describe a randomized optimization procedure, which combines the decision procedure with the approaches of Clarkson and Shor [13] and Chan [8]. The asymptotic expected running time of the optimization procedure is the same as that of the decision procedure.

Roughly speaking, Sharir's algorithm and its refinements [9, 16, 28] solve the decision problem for a given radius r by solving two separate subproblems, where in the first (resp., second) subproblem one wishes to determine whether P can be covered by two congruent balls of radius r so that the distance between their centers is at least r, (resp., at most r). We argue that if P can be covered by two balls of radius r then there exist two balls of radius r covering P such that the distance between their centers is at least δr, where $\delta = (2r_0/r)(1 - r/r_0)$. We extend the first part of Sharir's algorithm to three dimensions, so that it also works in the case where the distance between the centers of the two balls is at least δr (rather than r). However, we run into difficulties:

(i) A straightforward extension of Sharir's approach leads to a near-cubic algorithm, so we need additional geometric insights, described in Section 2 (see Lemma 2.2), to show that the algorithm can be implemented in near-quadratic time, when r^* is not too close to r_0.

(ii) We need an algorithm for the following problem: Given a sequence of m insertions and deletions of balls of fixed radius r into/from an initially empty set, let $\mathcal{B}_i$ be the set of balls after the first i update operations. We wish to determine, for each i, whether the common intersection of the balls in $\mathcal{B}_i$ is nonempty. Using the dynamic data structure for maintaining the 1-center in $\mathbb{R}^3$ [2], this can be accomplished in $O(m^\delta)$ time per update, for any $\delta > 0$. However, by exploiting the off-line nature of the problem and using multi-dimensional parametric searching, we design an off-line data structure that, after $O(m \log^2 m)$ preprocessing, can detect, for each i, whether $\bigcap \mathcal{B}_i \neq \emptyset$, in $O(\log^4 m \log\log m)$ time per update.

With the techniques just reviewed, the overall decision procedure can be implemented to run in nearly quadratic time (when r_0 is not too close to r^*).

The paper is organized as follows. Section 2 presents the decision algorithm, and Section 3 presents the optimization procedure. We describe the off-line ball-intersection-emptiness data structure in Section 4, and conclude in Section 5 with a few open problems.

2 The Decision Algorithm

Let P be a set of n points in $\mathbb{R}^3$ and $r > 0$ a parameter. We wish to determine whether P can be covered by two congruent balls of radius r. Let r_0 be the radius of the 1-center of P. We may assume that $r < r_0$ because otherwise the answer to the decision problem is obviously positive. For a point $x \in \mathbb{R}^3$, let $B(x)$ denote the ball of radius r centered at x. We call a pair of balls (B_1, B_2) of radius r a 2-*cover* of P if $P \subset B_1 \cup B_2$; it is called an *anchored* 2-cover if B_1 and B_2 cannot be brought closer to each other without losing a point of $P \cap B_1$ or $P \cap B_2$; see Figure 1 (a). We call P *coverable* if there exists a 2-cover of P. It is easy to verify that if P is coverable, then there is an anchored 2-cover of P, e.g., a 2-cover that minimizes the distance between the centers of its two balls is anchored.

For an oriented plane λ, let λ_L be the halfspace lying to the left of λ (i.e., in the direction of its negative normal). Let (B_1, B_2) be a 2-cover of P, and let c_1, c_2 be the respective centers of B_1 and B_2. Assume that c_1 lies to the left of c_2 (with respect to the normal direction of λ). We say that λ *weakly separates* B_1 and B_2 if the following three properties hold:

(P1) $c_1 \in \lambda_L$,

(P2) $P \cap \lambda_L \subset B_1$, and

(P3) $(P \cap \lambda_L) \cap \partial B_1 \neq \emptyset$.

If (B_1, B_2) is an anchored 2-cover, then they are, for example, weakly separated by the bisector plane π of c_1 and c_2. Indeed, $c_1 \in \pi_L$ and $P \cap \pi_L \subset B_1$. Furthermore, since (B_1, B_2) is an anchored cover, the hemisphere of B_1 bounded by the great circle parallel to π and lying to the left of c_1 must contain at least one point p of $P \cap \pi_L$. Hence, π weakly separates B_1 and B_2. We call a plane *weakly separating* if there is a 2-cover of P that is weakly separated by λ.

Guessing weakly separating planes. Following an argument similar to the one in [28], we show that there exists a small set of planes that contains at least one weakly separating plane for every anchored 2-cover of P, and that such a set can be constructed efficiently.

Lemma 2.1. *Let P be a set of n points in $\mathbb{R}^3$, and let $\beta(r) = 1/(1 - r/r_0)^3$ where r_0 is the radius of the 1-center ball of P. Then a set Λ of $O(\beta(r))$ planes can be computed in $O(n + 1/\beta(r))$ time so that every anchored 2-cover of P is weakly separated by at least one plane of Λ.*

Proof. Let (B_1, B_2) be an anchored 2-cover of P, and let c_1, c_2 be the centers of B_1, B_2, respectively. Suppose that $\|c_1c_2\| = \delta r$ for some parameter $\delta > 0$. Since the smallest ball enclosing B_1 and B_2, which is of radius $(1 + \delta/2)r$, also covers P, we conclude that $(1 + \delta/2)r \geq r_0$ or

$$\delta \geq 2(r_0/r - 1) = (2r_0/r)(1 - r/r_0).$$

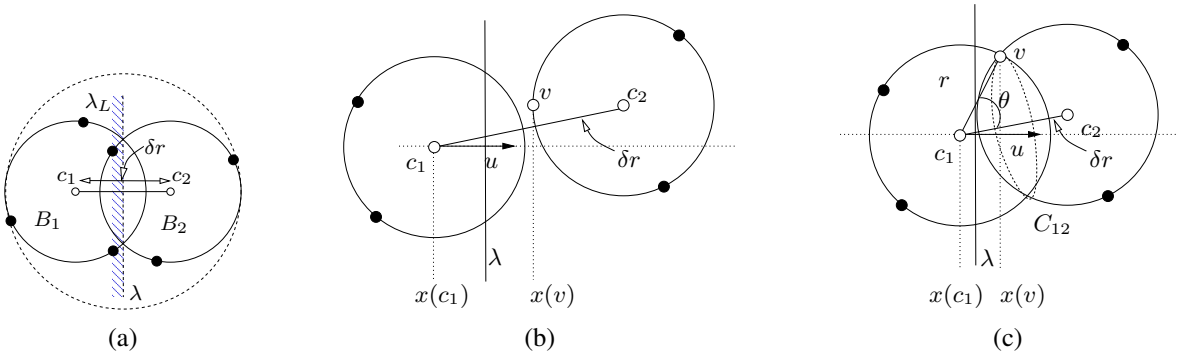


Figure 1. (a) Weakly separating plane λ. (b,c) Λ contains a weakly separating plane—(b) B_1 and B_2 are disjoint, and v is the leftmost point of B_2, (c) B_1 and B_2 intersect, and v is the leftmost point of their intersection circle.

We construct a set $\mathbb{D} \subset \mathbb{S}^2$ of directions in $\mathbb{R}^3$ so that the maximum angle between any direction u from its closest direction in $\mathbb{D}$ is at most $\alpha = \sin^{-1}((1 - r/r_0)/4)$. Since $\sin\alpha = (1 - r/r_0)/4$, we can construct, in $O(1/(1 - r/r_0)^2)$ time, a set $\mathbb{D}$ of size $O(1/(1 - r/r_0)^2)$ that satisfies the above property. Note that $\sin\alpha \leq \min\{1/4, \delta/8\}$. Fix a direction $u \in \mathbb{D}$, and let $I_u = [\min_{p\in P}\langle p, u\rangle, \max_{p\in P}\langle p, u\rangle]$; the length of I_u is at most $2r_0$. We choose a set Λ_u of parallel planes, each normal to direction u, that partition I_u into subintervals of length at most $\delta r/4$, i.e., the distance between two adjacent planes of Λ_u is at most $\delta r/4$. The size of Λ_u is at most $2r_0/(\delta r/4) = O(1/(1 - r/r_0))$. Set $\Lambda = \bigcup_{u\in\mathbb{D}} \Lambda_u$; we have $|\Lambda| = O(1/(1 - r/r_0)^3) = O(\beta(r))$.

Let us return to the anchored 2-cover (B_1, B_2). We claim that at least one plane of Λ weakly separates B_1 and B_2. Let $u \in \mathbb{D}$ be the direction closest to $\overrightarrow{c_1c_2}$. Rotate the coordinate frame so that u becomes the $(+x)$-direction. If B_1 and B_2 are disjoint, then let v be the leftmost point of B_2 (in the x-direction), otherwise let v be the leftmost point of the intersection circle of ∂B_1 and ∂B_2; see Figure 1 (b,c). We claim that $x(v) - x(c_1) \geq \delta r/4$.

First consider the case when B_1 and B_2 are disjoint, i.e., $\delta \geq 2$. The angle between $\overrightarrow{c_1c_2}$ and the x-axis is at most α and v is the leftmost point of B_2, so $x(v) - x(c_1) \geq \delta r\cos\alpha - r \geq \delta r/4$; the second inequality follows because $\delta \geq 2$ and $\sin\alpha \leq 1/4$ implies that $\cos\alpha \geq 3/4$. (See Figure 1 (b).) Assume then that B_1 and B_2 intersect. Let θ denote the angle $\measuredangle vc_1c_2$ (Figure 1 (c)). Using the triangle inequality on angles, the angle between $\overrightarrow{c_1v}$ and the x-axis is at most $\theta + \alpha$, so $x(v) - x(c_1) \geq r\cos(\theta + \alpha)$. Furthermore $\|c_1c_2\| = \delta r$, so $\cos\theta = \delta/2$. Hence,

$$x(v) - x(c_1) \geq r\cos(\theta + \alpha) = r(\cos\theta\cos\alpha - \sin\theta\sin\alpha).$$

Using the relations $\cos\theta = \delta/2$, $\sin\theta \leq 1$, $\cos\alpha \geq 3/4$, and $\sin\alpha \leq \delta/8$, we obtain

$$x(v) - x(c_1) \geq r\left(\frac{\delta}{2}\cdot\frac{3}{4} - \frac{\delta}{8}\right) = \frac{\delta r}{4}.$$

This proves the claim. Since the distance between any two adjacent planes of Λ_u is at most $\delta r/4$ and $[x(c_1), x(v)] \subseteq I_u$, at least one plane $\lambda \in \Lambda_u$ separates c_1 and v.

By construction $c_1 \in \lambda_L$ and $B_2 \cap \lambda_L \subset B_1$. Therefore $P \cap \lambda_L \subset B_1$. Since (B_1, B_2) is an anchored cover, the hemisphere of B_1 lying to the left of c_1, which is contained in λ_L, must contain at least one point of P. Hence, $(P \cap \lambda_L) \cap \partial B_1 \neq \emptyset$. Since this holds for any anchored 2-cover of P, the lemma follows. $\square$

For each $\lambda \in \Lambda$, we check whether it is a weakly separating plane. If the answer is yes, then we also compute a 2-cover of P that is weakly separated by λ. If none of the planes in Λ is weakly separating, then Lemma 2.1 and the preceding discussion imply that P is not coverable (by two balls of radius r). Before proceeding to describe in detail the implementation and analysis of this procedure, we note that the intersection of a finite collection $\mathcal{B}$ of congruent balls in $\mathbb{R}^3$ is a convex body bounded by spherical faces. As is well known (e.g., see [13]), each ball of $\mathcal{B}$ contributes at most one simply connected face to the boundary. The boundary is therefore a spherical map with at most $|\mathcal{B}|$ faces, so it has linear complexity. Moreover, one can construct the intersection in $O(|\mathcal{B}| \log |\mathcal{B}|)$ randomized expected time; see [13].

Reducing to a 2-dimensional search. We fix a plane $\lambda \in \Lambda$. Without loss of generality, we assume that λ is orthogonal to the x-axis. We describe an algorithm for testing whether λ is weakly separating. Let $P_L = P \cap \lambda_L$, $K = \bigcap_{p \in P_L} B(p)$, and $\sigma = \partial K \cap \lambda_L$. If $K \cap \lambda = \emptyset$ then $\sigma = \partial K$; otherwise σ has a "hole." Its boundary $\partial K \cap \lambda$, a convex piecewise-circular curve, is the boundary of the intersection of the disks $B(p) \cap \lambda$, for $p \in P_L$. If $\sigma = \emptyset$ then $K \cap \lambda_L = \emptyset$, so there is no ball of radius r that contains P_L and whose center lies in λ_L, and we can immediately conclude that λ cannot be a weakly separating plane. We thus assume that $\sigma \neq \emptyset$. By (P1) and (P3), $c_1 \in \sigma$, so it suffices to determine whether there is a point $\xi \in \sigma$ for which $P \setminus B(\xi)$ can be covered by a ball of radius r, i.e., whether $K(\xi) = \bigcap_{p \in P \setminus B(\xi)} B(p) \neq \emptyset$.

Let $P_R = P \setminus \lambda_L$. For a point $p \in P_R$, let $\gamma_p = \partial B(p) \cap \sigma$; γ_p is a curve consisting of at most $|P_L|$ circular arcs, bounding the unique face of $\partial K(P_L \cup \{p\}) \cap \partial B(p)$ within σ. Let $\Gamma = \{\gamma_p \mid p \in P_R\}$, and let $\mathbb{M}$ be the arrangement of Γ on σ, or rather its overlay with the planar map formed by the faces and edges of σ. By construction, for any point $p \in P$, $\partial B(p)$ does not cross the interior of any cell $\tau \in \mathbb{M}$, so either $\tau \subseteq B(p)$ or $\tau \cap B(p) = \emptyset$. For each cell (of any dimension) $\tau \in \mathbb{M}$, define $P_\tau = \{q \in P_R \mid \tau \cap B(q) = \emptyset\}$ and $K_\tau = \bigcap_{p \in P_\tau} B(p)$. For any point $x \in \tau$, $P \setminus B(x) = P_\tau$, as is easily verified, so $K(x) = K_\tau$. Hence, the problem of deciding whether λ is a weakly separating plane reduces to testing whether $K_\tau \neq \emptyset$ for some cell $\tau \in \mathbb{M}$.

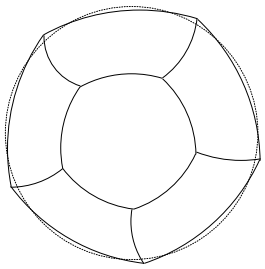


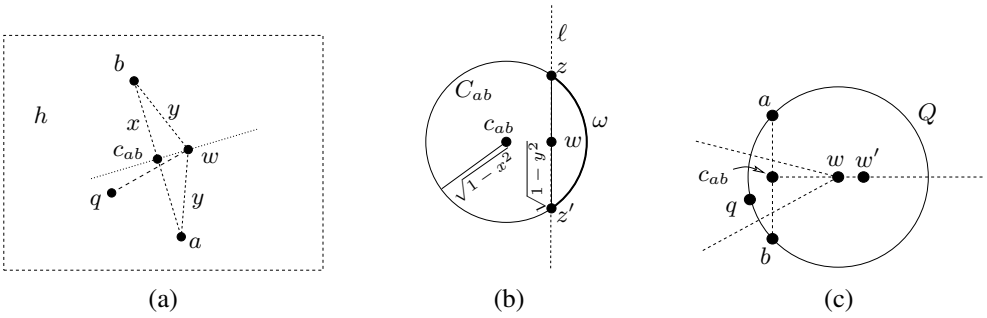
Figure 2. In a general setup (different from ours), an intersection circle of two balls (the dashed circle) may intersect a linear number of faces of $\partial K(P_L)$.

Complexity of $\mathbb{M}$. The running time of the algorithm is at least as large as the complexity of $\mathbb{M}$. Apriori, $\mathbb{M}$ might have cubic complexity, if many of the $O(n^2)$ pairs of curves γ_a, γ_b, for $a, b \in P_R$, traverse a linear number of common faces of σ, and intersect each other on many of these faces, in an overall linear number of points. Equivalently, the "danger" is that the intersection circle C_{ab} of a corresponding pair of spheres $\partial B(a), \partial B(b)$, for $a, b \in P_R$, could intersect a linear number of faces of σ (each of these intersections is an intersection point of the corresponding curves γ_a and γ_b). See Figure 2. Fortunately, we are able to show

that this cubic behavior is impossible in the assumed configuration—C_{ab} can meet only a constant number of faces of σ. Consequently, the overall complexity of $\mathbb{M}$ is only quadratic. The next lemma establishes the crucial property needed to prove a quadratic bound on the complexity of $\mathbb{M}$.

Lemma 2.2. *Let λ be a plane orthogonal to the x-axis, and put $P_L = P \cap \lambda_L$, $P_R = P \setminus P_L$. Let C_{ab} denote the intersection circle of $\partial B(a), \partial B(b)$, for some pair of points $a, b \in P_R$, and let $q \in P_L$. If the arc $\omega = C_{ab} \setminus B(q)$ is nonempty and smaller than a semicircle of C_{ab}, then at least one of its endpoints lies to the right of λ.*

Proof. The situation and its analysis are depicted in Figure 3. To slightly simplify the analysis, and without loss of generality, assume that $r = 1$. Let h be the plane passing through a, b and q. Let c_{ab} denote the midpoint of ab, and let w denote the center of the circumscribing circle Q of $\triangle qab$. Denote the distance $|ab|$ by $2x$, and the radius of Q by y (so $|wa| = |wb| = |wq| = y$). Note that c_{ab} and w lie in h and that $y \geq x$. Furthermore since $B(a), B(b)$, and $B(q)$ have radius 1 and their boundaries have nonempty intersection (consisting of the endpoints of w), we have $y \leq 1$. Observe that c_{ab} is the center of the intersection circle C_{ab}. See Figure 3 (a).



(a) (b) (c)

Figure 3. The setup in Lemma 2.2: (a) the setup within the plane h; (b) the setup within C_{ab}; (c) ww' lies on the bisector of ab in the direction that gets away from q.

The intersection points z, z' of C_{ab} and $\partial B(q)$ are the intersection points of the three spheres $\partial B(a)$, $\partial B(b)$, and $\partial B(q)$. They lie on the line ℓ passing through w and orthogonal to h, at equal distances $\sqrt{1-y^2}$ from w. See Figure 3 (b). Hence, within C_{ab}, zz' is a chord of length $2\sqrt{1-y^2}$. In the assumed setup, z and z' delimit a short arc ω of C_{ab}, which lies outside $B(q)$, so points on the arc are (equally) closer to a and b than to q.

Hence, the projection of the arc ω onto h is a small interval ww', which lies on the bisector of ab in the direction that gets away from q; that is, it lies on the Voronoi edge of ab in the diagram $\mathrm{Vor}(\{a, b, q\})$ within h. See Figure 3 (c). Moreover, c_{ab} also lies on the bisector, but it has to lie on the other side of w, or else the smaller arc ω would have to lie inside $B(q)$. That is, c_{ab} has to be closer to q than to a and b. Since λ separates $a, b \in P_R$ from $q \in P_L$, it also separates c_{ab} from q. Moreover, the preceding arguments are easily seen to imply that wq crosses ab (as in Figure 3 (a)), which implies that λ also separates q and w, so w has to lie to the right of λ. Since z and z' lie on two sides of w on the line ℓ, at least one of them has to lie on the same side of λ as w (i.e., to the right of λ). This completes the proof. □

Lemma 2.3. *The complexity of $\mathbb{M}$ is $O(n^2)$.*

Proof. Let $a, b \in P_R$ and consider those arcs of the intersection circle $C_{ab} = \partial B(a) \cap \partial B(b)$ which lie outside K but whose endpoints lie on σ. Clearly, all these arcs are pairwise disjoint. At most one such arc can be larger than a semicircle. Let ω be an arc of this kind which is smaller than a semicircle, and let $q \in P_L$ be such that (at least) one endpoint of ω lies on $\partial B(q)$. Then $\omega' = C_{ab} \setminus B(q) \subseteq \omega$ is also smaller

than a semicircle. By Lemma 2.2, exactly one endpoint of ω' lies to the right of λ (the other endpoint lies on σ). Note that C_{ab} cannot have more than two such short arcs lying outside K, since, due to the convexity of C_{ab}, only two arcs of C_{ab} can have their two endpoints lying on opposite sides of λ. Hence the number of arcs of C_{ab} under consideration is at most three (one long and two short), implying that γ_a and γ_b intersect at most six times, and thus the complexity of $\mathbb{M}$ is $O(n^2)$, as asserted. □

Constructing and searching $\mathbb{M}$. The next step of the algorithm is to compute $\mathbb{M}$. As already noted, ∂K has linear complexity and can be computed in $O(n \log n)$ randomized expected time [13], and σ can be computed in additional linear time. We then compute the intersection curve γ_p of $B(p)$ and σ, for each $p \in P_R$, in $O(n)$ time, by computing the intersection of $\partial B(p)$ with each face of σ. Set $\Gamma = \{\gamma_p \mid p \in P_R\}$. For each face $f \in \sigma$, let Γ_f be the collection of subarcs of Γ that lie in f. The total cost of computing Γ and Γ_f for all faces f of σ is $O(n^2)$. Next, for each face $f \in \sigma$, we compute the arrangement of Γ_f, using a standard sweep-line algorithm [14], in time $O((n + k_f) \log n)$ time, where k_f is the complexity of the arrangement of Γ_f. By Lemma 2.3, $\sum_{f \in \mathbb{M}} k_f = O(n^2)$, so the total cost of computing $\mathbb{M}$ is

$$\sum_{f \in \sigma} O((n + k_f) \log n) = O(n^2 \log n).$$

Let $\mathbb{M}^*$ be the dual graph of $\mathbb{M}$—each node of $\mathbb{M}^*$ corresponds to a two-dimensional face of $\mathbb{M}$, and (τ_1, τ_2) is an edge in $\mathbb{M}^*$ if the faces τ_1 and τ_2 share an edge e in $\mathbb{M}$. If $e \subseteq \gamma_p$, then $P_{\tau_1} \oplus P_{\tau_2} = \{p\}$, and we label the edge $\{\tau_1, \tau_2\}$ with p. If e is a portion of an original edge of σ, $P_{\tau_1} = P_{\tau_2}$ and no labeling is needed. We compute a path $\Pi = \langle \tau_1, \ldots, \tau_u \rangle$ in $\mathbb{M}^*$, for $u = O(|\mathbb{M}|)$, that visits each node of $\mathbb{M}^*$ at least once (see, e.g., [1]). It suffices to check whether $K_{\tau_i} \neq \emptyset$ for some $1 \leq i \leq u$.[1] As just observed, for any $1 \leq i < u$, either $P_{\tau_i} \oplus P_{\tau_{i+1}} = \{p_i\}$ for some $p_i \in P_R$, or $P_{\tau_{i+1}} = P_{\tau_i}$ if τ_i, τ_{i+1} are separated by an edge of σ. Therefore $K_{\tau_{i+1}}$ can be constructed from K_{τ_i} by inserting or deleting the ball $B(p_i)$. We thus need an algorithm for the following problem: Given an initial set $\mathcal{B}$ of balls of radius r and a sequence of u insertions/deletions of balls of radius r into/from $\mathcal{B}$, let K_i be the intersection of the balls in $\mathcal{B}$ after the first i update operations. The problem is to determine whether there exists an i for which $K_i \neq \emptyset$.

Using the dynamic data structure of Agarwal *et al.* [2], the emptiness of the intersection of congruent balls in $\mathbb{R}^3$ can be detected in $O(n^\varepsilon)$ time, for any $\varepsilon > 0$, per update, thereby leading to an $O(n^{2+\varepsilon})$-time algorithm for determining whether some K_i is nonempty. However, the sequence of updates in our case is given in advance, so we can do better using the approach of Edelsbrunner and Overmars [15] for off-line dynamic problems. In our context, their approach transforms the problem of emptiness-detection of the intersection of balls after each update to the following query problem: Let $\mathbb{B} = \{\mathcal{B}_1, \ldots, \mathcal{B}_m\}$, $m = O(u) = O(n^2)$, where each $\mathcal{B}_i$ is a collection of balls of radius r in $\mathbb{R}^3$, and $N = \sum_i |\mathcal{B}_i| = O(n^2 \log n)$. Set $K_i = \bigcap \mathcal{B}_i$, for $1 \leq i \leq m$. The task is to preprocess $\mathbb{B}$ into a data structure so that, for a query subset $J \subseteq [1, \ldots, m]$, one can quickly determine whether $\bigcap_{j \in J} K_j \neq \emptyset$; the number of queries asked is $u = O(n^2)$; see the original paper [15] for details.

We present in Section 4 (cf. Corollary 4.2) an algorithm that preprocesses $\mathbb{B}$ in $O(N \log N) = O(n^2 \log^2 n)$ time so that, for a query subset J, one can determine, in $O(\log |J| \log^4 n)$ randomized expected time, whether $\bigcap_{j \in J} K_j \neq \emptyset$. Plugging this algorithm into the Edelsbrunner-Overmars framework, we can determine, in a total of $O(n^2 \log^4 n \log \log n)$ time, whether $K_i \neq \emptyset$ for some $1 \leq i \leq u$. Putting everything together we obtain the following result.

Theorem 2.4. *Let P be a set of n points in $\mathbb{R}^3$, and let $r > 0$ be a parameter. We can determine in $O(\beta(r) n^2 \log^4 n \log \log n)$ randomized expected time whether P can be covered by two balls of radius r each, where $\beta(r) = 1/(1 - r/r_0)^3$ and r_0 is the radius of the smallest ball containing P.*

[1] Actually, this only checks whether $K_\tau \neq \emptyset$ for some 2-face τ of $\mathbb{M}$. The procedure, however, can be easily adapted to also check whether $K_\tau \neq \emptyset$ for some edge or vertex τ of $\mathbb{M}$. We omit these details from here.

3 The Optimization Procedure

This section describes a randomized algorithm for computing the 2-center of P, by combining the decision procedure described above with the randomized technique used by Clarkson and Shor [13] (see also Chan [8]).

Let r_0 and r^* be the radii of the 1-center and the 2-center of P, respectively. We first compute r_0 in linear time [11]. We then consider the interval $(0, r_0)$, which contains r^*, and run an *exponential search* through it, calling the decision procedure with the values $r_i = r_0(1-1/2^i)$, for $i = 1, 2, \ldots$, in order, until the first time we reach a value $r' = r_i \geq r^*$ for which the decision procedure returns YES. Note that $1 - r'/r_0 = 1/2^i$ and $1/2^i < 1 - r^*/r_0 < 1/2^{i-1}$, so $\beta(r') = O(\beta(r^*)) = \Theta\left(2^{3i}\right)$. Moreover, the sequence $\{\beta(r_i)\}_{i\geq 1}$ forms a geometric series, so the overall cost of the exponential search is $O(\beta(r^*)n^2 \log^4 n \log\log n)$. All further calls to the decision procedure will be made for values $r \in (r'/2, r')$ (clearly, r^* lies in that interval), implying that $\beta(r) = O(\beta(r^*))$ for all such calls.[2]

Let B_1 and B_2 be two balls of radius r^*, centered at c_1 and c_2 respectively, that cover P. Let $P_1 = \{p \in P \mid \|c_1 p\| \leq \|c_2 p\|\}$ and $P_2 = P \setminus P_1$. Then $P_1 \subset B_1$, $P_2 \subset B_2$, and P_1, P_2 are separated by the bisector plane of c_1 and c_2. This suggests the following straightforward but inefficient approach for computing a 2-center of P: We consider all $O(n^3)$ distinct (equivalence classes of) planes that separate P into two nonempty subsets. For each such plane π, we compute $r_0(P \cap \pi^+)$ and $r_0(P \cap \pi^-)$, where π^+ (resp., π^-) is the halfspace lying above (resp., below) π and $r_0(X)$ is the radius of the smallest ball enclosing a set X, and set r_π to be the larger of the two radii. We return the two balls for which r_π is minimized (inflating the smaller ball, if needed, to make them congruent). A naive implementation of this approach takes $O(n^4)$ time (and skips altogether the decision procedure of the previous section). We show how this approach can be refined so that we neither process all $O(n^3)$ planes nor do we spend linear time for each plane that we process.

We use a recursive procedure for computing r^*. A recursive subproblem consists of a partition of P into three subsets P^+, P^-, Q and a current upper bound $r_{\text{curr}} \in [r^*, r']$. The procedure returns the smallest radius $\varrho = \varrho(P^+, P^-, Q)$, for which there are two congruent balls B^+ and B^- of radius ϱ, centered at two respective centers c^+ and c^-, such that (i) $P \subset B^+ \cup B^-$, $P^+ \subset B^+$ and $P^- \subset B^-$, and (ii) c^+ lies above the bisector plane of c^+, c^- (and c^- lies below it). The procedure also returns some 2-cover (B^+, B^-) that satisfies these properties, which we denote by $\mathcal{B}(P^+, P^-, Q)$. It maintains the invariant that if a recursive call is made with $(P^+, P^-, Q, r_{\text{curr}})$, then $\varrho(P^+, P^-, Q) \leq r_{\text{curr}}$. Initially, $P^+ = P^- = \emptyset$, $Q = P$, and $r_{\text{curr}} = r'$, and the invariant clearly holds. Let $\mathcal{B}_{\text{curr}}$ be the best 2-cover computed so far by the recursive procedure; initially $\mathcal{B}_{\text{curr}}$ is the 2-cover corresponding to the radius r'.

We show below that the decision procedure described in Section 2 can be adapted to determine, in $O(\beta(r_{\text{curr}})|Q|n \log^4 n \log\log n)$ time, whether $\varrho(P^+, P^-, Q) \leq r_{\text{curr}}$ for a given partition (P^+, P^-, Q) of P; only if the answer is YES we call our recursive procedure to compute $\varrho(P^+, P^-, Q)$, thereby ensuring that the invariant is maintained. For now let us assume that such a modified version of the procedure exists.

We need the standard notion of $(1/t)$-cuttings, specialized to three dimensions: Given a collection H of n planes in $\mathbb{R}^3$ and a parameter $1 \leq t \leq n$, a $(1/t)$-*cutting* of H is a partition of $\mathbb{R}^3$ into (possibly unbounded) pairwise openly disjoint simplices such that the interior of each simplex is intersected by at most n/t planes of H. It is well known that a $(1/t)$-cutting of H of size $O(t^3)$ exists, and that it can be computed in $O(nt^2)$ time [10]. Given a simplex $\triangle$ and a set S of points, let $S_\triangle, S^+_\triangle, S^-_\triangle \subseteq S$ be the subsets of S, consisting of those points whose dual planes intersect $\triangle$, lie above $\triangle$, and lie below $\triangle$, respectively.

Let $(P^+, P^-, Q, r_{\text{curr}})$ be a recursive subproblem. For a point $x \in \mathbb{R}^3$, let x^* denote the plane dual to x; set $Q^* = \{q^* \mid q \in Q\}$. We fix a sufficiently large constant $t > 1$. If $|Q| \leq t$, we compute

[2] These initial stages of the algorithm ensure that we do not call the decision procedure with values of r too close to r_0, thereby ensuring that it is not less efficient than it really needs to be.

$\varrho(P^+, P^-, Q)$ (and $\mathcal{B}(P^+, P^-, Q)$) in $O(n)$ time, by considering all $O(t^3)$ linearly separable partitions of Q into two subsets Q^+, Q^-, computing $\max\{r_0(P^+ \cup Q^+), r_0(P^- \cup Q^-)\}$ for each such partition, and returning the partition that attains the smallest value among these radii. So assume that $|Q| > t$. We compute a $(1/t)$-cutting $\Xi = \{\triangle_1, \ldots, \triangle_u\}$ of $Q^\star$, where $u \leq ct^3$ and c is a constant independent of t. We process the simplices of Ξ in a *random* order. For each $i \leq u$, using the (adapted) decision procedure we determine whether $\varrho(P^+ \cup Q^+_{\triangle_i}, P^- \cup Q^-_{\triangle_i}, Q_{\triangle_i}) \leq r_{\text{curr}}$. If the answer is NO, we move to the next simplex in the random permutation of Ξ. Otherwise, we recursively solve the subproblem $(P^+ \cup Q^+_{\triangle_i}, P^- \cup Q^-_{\triangle_i}, Q_{\triangle_i}, r_{\text{curr}})$. Let ϱ and $\mathcal{B}$ be the radius and the 2-cover, respectively, returned by the recursive call. We set $r_{\text{curr}} = \varrho$ and $\mathcal{B}_{\text{curr}} = \mathcal{B}$. After processing all the simplices of Ξ, we return r_{curr} and $\mathcal{B}_{\text{curr}}$ as the respective values of $\varrho(P^+, P^-, Q)$ and $\mathcal{B}(P^+, P^-, Q)$.

The correctness of the algorithm follows from the following lemma.

Lemma 3.1. $\varrho(P^+, P^-, Q) = \min_{\triangle \in \Xi} \varrho(P^+ \cup Q^+_\triangle, P^- \cup Q^-_\triangle, Q_\triangle)$.

Proof. Clearly, the right-hand side is an upper bound on $\varrho(P^+, P^-, Q)$. To show equality, let B^+ and B^- be the balls constituting an optimal 2-cover of P, with respective centers c^+, c^- , such that $P^+ \subset B^+$, $P^- \subset B^-$, and c^+ lies above the bisector plane π of c^+ and c^-; here optimality is with respect to the collection of 2-covers that satisfy these properties. Let $Q^+ = \{q \in Q \mid \|qc^+\| \leq \|qc^-\|\}$, and put $Q^- = Q \setminus Q^+$. Then $Q^+ \subset B^+$ and $Q^- \subset B^-$. Suppose that the point dual to π lies in a simplex $\triangle$ of Ξ. Then $Q^+_\triangle \subseteq Q^+$ and $Q^-_\triangle \subseteq Q^-$. Therefore $\varrho(P^+, P^-, Q) = \varrho(P^+ \cup Q^+_\triangle, P^- \cup Q^-_\triangle, Q_\triangle)$, and the lemma follows. □

The analysis in Clarkson and Shor [13] (or in Chan [8]) implies that when we execute our procedure on a subproblem $(P^+, P^-, Q, r_{\text{curr}})$, it makes recursive calls at an expected number of $\ln|\Xi| = \ln(ct^3)$ simplices of Ξ. Since Ξ is a $(1/t)$-cutting, $|Q_\triangle| \leq |Q|/t$ for each simplex $\triangle \in \Xi$. Furthermore, the algorithm spends $O(\beta(r^*)|Q_\triangle| n \log^4 n \log\log n)$ time to determine whether $\varrho(P^+ \cup Q^+_\triangle, P^- \cup Q^-_\triangle, Q_\triangle) \leq r_{\text{curr}}$ for each $\triangle \in \Xi$ (recall that $\beta(r_{\text{curr}}) = O(\beta(r^*))$). The total cost of these tests is thus $O(\beta(r^*)t^3(|Q|/t)n \log^4 n \log\log n) = O(\beta(r^*)|Q| n \log^4 n \log\log n)$ (recalling that t is a constant). Let $T(m)$ be the expected running time for a subproblem with $|Q| = m$. Then we obtain the following recurrence (in which n is regarded as a fixed "global" quantity).

$$T(m) \leq \begin{cases} \ln(ct^3)T(m/t) + O\left(\beta(r^*)mn \log^4 n \log\log n\right), & \text{for } m > t, \\ O(n), & \text{for } m \leq t, \end{cases} \tag{1}$$

where c is the constant in the bound on the size of the cutting. If we choose t sufficiently large so that $t \geq 2\ln(ct^3)$, then, as can easily be checked, the solution to the above recurrence is

$$T(m) = O(\beta(r^*)mn \log^4 n \log\log n).$$

Since $m = n$ for the initial subproblem, the overall optimization procedure runs in $O(\beta(r^*)n^2 \log^4 n \log\log n)$ expected time.

Adapting the decision procedure. Let P^+, P^-, Q be a partition of P, with $|Q| = m$, and $r > 0$ a parameter. We wish to determine whether $\varrho(P^+, P^-, Q) \leq r$. That is, we wish to determine whether there exists a 2-cover (B_1, B_2) of P, with the center of B_2 lying above the bisector plane of the centers of the two balls, for which

(P4) $P^- \subset B_1$ and $P^+ \subset B_2$.

Note that if there is a 2-cover of P satisfying (P4), then there is also an anchored 2-cover with the same property (including the locations of the ball centers with respect to their bisector plane). This follows by observing that the two balls can be moved toward each other to make the cover anchored so that neither of the balls loses any point that it already contains. Using Lemma 2.1, we compute a family Λ of $O(\beta(r))$ planes such that any anchored 2-cover of P is weakly separated by at least one of the planes of Λ. We fix a plane $\lambda \in \Lambda$ and test whether there exists a 2-cover of P satisfying (P1)–(P3) (as listed earlier) as well as (P4). For simplicity, let us assume that λ is orthogonal to the x-axis, and, with a slight abuse of terminology, regard the right halfspace λ_R (resp., left halfspace λ_L) as lying above (resp., below) λ.

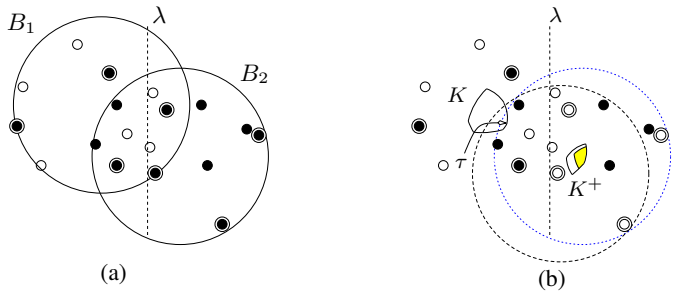


Figure 4. (a) The sets P^- (hollow circles), P^+ (filled circles), and Q (double circles); a 2-cover of P satisfying (P1)–(P4). (b) The sets K, K^+, and $K^+ \cap K_\tau$ (shaded portion of K^+); a pair of circles (drawn as dashed) centered at a pair of points of Q_R (double hollow circles) that intersect K.

Put $Q_L = Q \cap \lambda_L$ and $Q_R = Q \cap \lambda_R$. If λ is indeed weakly separating, Q_L must be contained in B_1, and so does P^-, by (P4). We thus form $K = \bigcap_{p \in P^- \cup Q_L} B(p)$, and consider $\sigma = \partial K \cap \lambda_L$. As before, K and σ can be computed in $O(n \log n)$ randomized expected time [13]. By (P4), all the points of P^+ have to be contained in B_2, so its center c_2 must be contained in $K^+ = \bigcap_{p \in P^+} B(p)$. The only "undecided" points are those of Q_R. We thus consider, for each $p \in Q_R$, the curve $\gamma_p = \partial B(p) \cap \sigma$, and set $\Gamma = \{\gamma_p \mid p \in Q_R\}$. See Figure 4. Let $\mathbb{M}$ be the arrangement of Γ on σ (including, as earlier, the edges of $\partial\sigma$ and their interaction with the curves of Γ). The analysis in the proof of Lemma 2.2 can easily be extended to yield the following variant.

Lemma 3.2. *The complexity of $\mathbb{M}$ is $O(mn + m^2) = O(mn)$.*

Arguing as before, the total time spent in computing $\mathbb{M}$ is $O(mn \log n)$. As earlier, we need to test each face $\tau \in \mathbb{M}$ for a possible location of the center c_1 of B_1. Put $Q_\tau = \{q \in Q_R \mid B(q) \cap \tau = \emptyset\}$, and set $K_\tau = \bigcap_{q \in Q_\tau} B(q)$. We note that τ is a valid location for c_1 if and only if $K_\tau \cap K^+ \neq \emptyset$ (any point in this intersection is a valid location for c_2). We therefore proceed in the same way as in Section 2, to test for the existence of a face τ for which $K_\tau \cap K^+ \neq \emptyset$, with the following modifications. First, the size of the update sequence is now $O(mn)$. Second, each of the intersections that we examine has to contain the set K^+. We thus begin with precomputing $K^+ = \bigcap_{q \in P^+} B(q)$, in $O(n \log n)$ randomized expected time, and then proceed, following the same algorithm and analysis as before, to determine, in a total of $O(mn \log^4 n \log\log n)$ expected time, whether $K_\tau \cap K^+ \neq \emptyset$ for some $\tau \in \mathbb{M}$. If such a nonempty intersection is detected, we conclude that $\rho(P^+, P^-, Q) \leq r$. We repeat this procedure for all planes $\lambda \in \Lambda$. If the answer is negative for all planes in Λ, we conclude that $\rho(P^+, P^-, Q) > r$.

Putting everything together, we obtain the main result of the paper.

Theorem 3.3. *Let P be a set of n points in $\mathbb{R}^3$. A 2-center for P can be computed in $O(\beta(r^*)n^2 \log^4 n \log\log n)$ randomized expected time, where $\beta(r) = 1/(1 - r_0/r)^3$, r_0 is the radius of the smallest enclosing ball of P, and $r^* \leq r_0$ is the radius of the balls of the 2-center for P.*

4 Emptiness Detection of Intersection of Congruent Balls

We now describe an efficient procedure for testing emptiness of the intersection of congruent balls in three dimensions. This procedure can easily be modified so that it can determine, in cases where the intersection is nonempty, whether it is degenerate (i.e., contains only a single point), or, more generally, compute the (unique) leftmost point in the intersection (the point with the smallest x-coordinate). Determining degeneracy is important in identifying the case in which the input radius r of the decision procedure is equal to the optimal radius r^*. Finding the leftmost point in the intersection will be useful for a slight speed-up of the procedure, discussed at the end of this section.

For a set $\Upsilon = \{\gamma_1, \ldots, \gamma_m\}$ of geometric regions in $\mathbb{R}^3$, let $\mathcal{I}(\Upsilon) = \bigcap_{i=1}^m \gamma_i$ denote their intersection. Let $\mathcal{B}$ be a set of n congruent balls in $\mathbb{R}^3$, of some radius r, and let $\mathbb{B} = \{\mathcal{B}_1, \ldots, \mathcal{B}_m\}$ be a collection of m subsets $\mathcal{B}_i \subseteq \mathcal{B}$, for $1 \leq i \leq m$. Set $n_i = |\mathcal{B}_i|$ and $N = \sum_i n_i$. We wish to preprocess $\mathbb{B}$ into a data structure so that for any $J \subseteq \{1, \ldots, m\}$ we can determine, in $O(|J| \log^4 n)$ time, whether $\bigcap_{j \in J} \mathcal{I}(\mathcal{B}_j) \neq \emptyset$. If the answer is YES, then we also wish to return a "witness" point in the intersection. To simplify the forthcoming analysis, we assume that the centers of the balls in $\mathcal{B}$, which in our setup are the input points of P, are in general position. In particular, this means that no five of them are co-spherical, and that, for any fixed radius, there exists at most one quadruple of centers lying on a common sphere of that radius. In particular, this holds for the specific radius r under consideration. Additional properties of the assumed general position are noted later on in the analysis. The edges and vertices of $\mathcal{I}(\mathcal{B}_i)$ are the intersections of two or three bounding spheres, respectively (by assumption, at most one vertex might be incident on four spheres). As noted earlier, each ball $B \in \mathcal{B}_i$ contributes at most one (simply connected) face to $\partial\mathcal{I}(\mathcal{B}_i)$ (see [13]). Hence, $\partial\mathcal{I}(\mathcal{B}_i)$ is a planar (or, rather, spherical) map of $O(n_i)$ size, and it can be constructed in $O(n_i \log n_i)$ randomized expected time (see [13]).

It will be convenient to represent each ball $B \in \mathcal{B}$ as the intersection of two *quasi-cylinders* B^+ and B^-, defined as follows. Let c, r be the center and radius of B, respectively, and let $\lambda = \{(0, 0, z) \mid z \geq 0\}$ be the ray emanating from $(0, 0, 0)$ in the $(+z)$-direction. We define $B^+ = B \oplus \lambda$ and $B^- = B \oplus (-\lambda)$; ∂B^+ consists of the lower hemisphere of B plus a semi-infinite vertical cylinder of radius r whose axis is $c \oplus \lambda$, and ∂B^- consists of the upper hemisphere of B plus a semi-infinite vertical cylinder of radius r with axis $c \oplus (-\lambda)$. We refer to these two types of quasi-cylinders as *upper quasi-cylinders* and *lower quasi-cylinders*, respectively. See Figure 5 (a). Clearly $B = B^+ \cap B^-$. Put $\mathcal{G} = \{B^+, B^- \mid B \in \mathcal{B}\}$. For each $i \leq m$, let $\mathcal{B}_i^+ = \{B^+ \mid B \in \mathcal{B}_i\}$ and $\mathcal{B}_i^- = \{B^- \mid B \in \mathcal{B}_i\}$. Then $\bigcap_{j \in J} \mathcal{I}(\mathcal{B}_j) = \bigcap_{j \in J} (\mathcal{I}(\mathcal{B}_j^+) \cap \mathcal{I}(\mathcal{B}_j^-))$. Each of $\partial\mathcal{I}(\mathcal{B}_i^+)$ and $\partial\mathcal{I}(\mathcal{B}_i^-)$ is a planar map of size $O(n_i)$. We can thus reformulate the original problem as follows: Let $\mathbb{G} = \{\mathcal{G}_1, \ldots, \mathcal{G}_{2m}\}$ where $\mathcal{G}_{2i-1} = \mathcal{B}_i^+$ and $\mathcal{G}_{2i} = \mathcal{B}_i^-$ for $i = 1, \ldots, m$. Set $K_j = \mathcal{I}(\mathcal{G}_j)$ for $j = 1, \ldots, 2m$, and put $\mathcal{K} = \{K_1, \ldots, K_{2m}\}$. We seek a procedure that, given any $J \subseteq \{1, \ldots, 2m\}$, can detect, in $O(|J| \log^4 n)$ time, whether $K = \bigcap_{j \in J} K_j \neq \emptyset$; if $K \neq \emptyset$, then it should also return a point in K.[3]

For a point $z \in \mathbb{R}^3$, let $z^\downarrow$ denote its xy-projection, and for a set $S \subset \mathbb{R}^3$, let $S^\downarrow = \{z^\downarrow \mid z \in S\}$ denote the xy-projection of S. For a set $\mathcal{H} \subseteq \mathcal{G}$, let $\mathcal{I}^\downarrow(\mathcal{H})$ denote the xy-projection of the corresponding intersection $\mathcal{I}(\mathcal{H})$. For each $i = 1, \ldots, 2m$, let M_i denote the xy-projection of ∂K_i, which is a planar subdivision: The vertical boundary of ∂K_i projects to a piecewise-circular closed convex curve bounding M_i (all these circles have radius r). The other (interior) edges of M_i are straight segments and elliptic arcs (see Figure 5 (b)). If an edge of M_i is not x-monotone, we split it into two or three x-monotone subarcs by adding a vertex or a pair of vertices at the points of its y-tangency. Set $\mathcal{M} = \{M_1, \ldots, M_{2m}\}$. Let X denote the sequence of the x-coordinates of the vertices of all the maps in $\mathcal{M}$, sorted in increasing order. For a point $p \in \mathbb{R}^2$, let $p^\uparrow$ be the line passing through p in the z-direction. For each $i \leq m$, if $p^\uparrow \cap K_i \neq \emptyset$ (i.e.,

[3] In our application, we only use sets J that contain $2i$ if and only if they contain $2i - 1$, for $i = 1, \ldots, m$. Nevertheless, the procedure will not make use of this additional property.

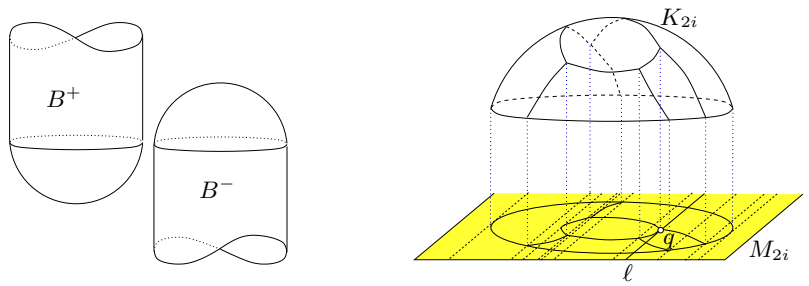


Figure 5. (a) Quasi-cylinders B^+ and B^-. (b) ∂K_{2i}, its xy-projection M_{2i}, a vertex q of M_{2i}, and a y-parallel line passing through q.

$p \in K_i^{\downarrow}$), we set $\varrho_i(p)$ to $K_i \cap p^{\uparrow}$, which is a ray in the $(-z)$-direction (resp., $(+z)$-direction) if i is even (resp., odd); otherwise $\varrho_i(p)$ is undefined. If $p \in K_i^{\downarrow}$ then we define $\Omega_i(p)$ to be the set of (at most four) quasi-cylinders in $\mathcal{G}_i$ (of the same type) whose boundaries pass through the endpoint of $\varrho_i(p)$; otherwise, we can obtain a set of (at most) four quasi-cylinders in $\mathcal{G}_i$ whose intersection does not contain $p^{\uparrow}$ (e.g., by picking a point $q \in K_i$ nearest to $p^{\uparrow}$ and collecting the at most four quasi-cylinders whose boundaries pass through q), and we set $\Omega_i(p) \subseteq \mathcal{G}_i$ to the collection of these "witness" quasi-cylinders.[4]

Preprocessing step. For each $i = 1, \ldots, 2m$, we compute K_i and M_i. We preprocess M_i using the standard point-location algorithm of Sarnak and Tarjan [27]. That is, we partition M_i into slabs by y-parallel lines through the vertices of M_i (recalling that all the edges of M_i are assumed to be x-monotone) and store the sequence of edges of M_i intersecting each slab in their y-sorted order, so that each of the following operations can be performed in $O(\log n_i) = O(\log n)$ time:

(A1) For a point $p \in \mathbb{R}^2$, return $\varrho_i(p)$ and $\Omega_i(p)$.

(A2) For a y-interval δ, count the number of edges of M_i that intersect δ, or return a random edge among them.

(A3) For an elliptic arc (or a straight segment) γ that lies within a single slab, count the number of edges of M_i that intersect γ, or return a random edge among them.

Using the persistent data structure of Sarnak and Tarjan [27], M_i can be preprocessed in $O(n_i \log n_i)$ time into a data structure of $O(n_i \log n_i)$ size so that each operation of the types (A1)–(A3) can be performed in $O(\log n_i)$ time.[5] Operations of type (A1) or (A2) are straightforward to implement. Performing an operation of type (A3) with an arc e can be done using a binary search through the y-structure of the slab, where in each step we check, in $O(1)$ time, whether e intersects some arc of the y-structure. Since the sequence of edges intersecting γ is a contiguous portion of the y-structure, the procedure just sketched works correctly. More precisely, the binary search returns the portion of the y-structure intersected by the query segment or arc, as the disjoint union of logarithmically many canonical subsets. Using this representation, a random edge of this portion can be drawn in a straightforward and standard manner. Hence, the overall size and preprocessing time of the data structure are both $O(N \log n)$.

[4] Generally, only three quasi-cylinders are needed to form a witness set. We use the bound four to cater to the potential unique vertex incident to four boundaries.

[5] The size of the data structure is $O(n_i \log n_i)$ instead of $O(n_i)$ because of the counting queries in (A2) and (A3); see the original paper [27] for details.

Sketch of the query procedure. Fix a set $J \subset \{1, \ldots, 2m\}$; set $|J| = t$. Put $K = \bigcap_{j \in J} K_j$, $\mathcal{G}_J = \bigcup_{j \in J} \mathcal{G}_j$, and let E be the union of the sets of edges in the maps M_j, over all $j \in J$. We describe a procedure that detects whether $K^{\downarrow} \neq \emptyset$. If the answer is YES, it also returns a point $\xi \in K$.

Before giving an outline of the procedure we make two observations, which will be useful for our algorithm.

(C1) Let $\psi \subset \mathbb{R}^2$ be a set of points (e.g., a single point, a line segment, an arc, or even a two-dimensional region) that lies inside a single face (of any dimension), say, f_j, of each M_j, for $j \in J$. Set $\mathcal{G}_{|\psi} = \bigcup_{j \in J} \Omega_j(f_j)$ (where $\Omega_j(f_j)$ is a shorthand notation for the common value $\Omega_j(p)$, for every $p \in f_j$) and $K_{|\psi} = \mathcal{I}(\mathcal{G}_{|\psi})$. Then $|\mathcal{G}_{|\psi}| = O(t)$, $K^{\downarrow} \cap \psi = K_{|\psi}^{\downarrow} \cap \psi$, and for each $p \in \psi$, $p^{\uparrow} \cap K = p^{\uparrow} \cap K_{|\psi}$. Moreover, $K_{|\psi}$ can be computed in $O(t \log n)$ time.

(C2) Let ψ be a point or a line in $\mathbb{R}^2$ and let $\mathcal{H} \subset \mathcal{G}$ be a set of quasi-cylinders. If $\mathcal{I}^{\downarrow}(\mathcal{H}) \cap \psi = \emptyset$, then one can compute, in $O(|\mathcal{H}| \log |\mathcal{H}|)$ time, a *witness* set $\Omega_\psi \subset \mathcal{H}$ of size at most 4, such that $\mathcal{I}^{\downarrow}(\Omega_\psi) \cap \psi = \emptyset$.

(C1) is straightforward, and (C2) follows from Helly's theorem and other known results (see, e.g., [23]).

We use a multi-dimensional prune-and-search technique similar to the one in Matoušek [23] (see also [2, 7]). We solve in succession the following three subproblems; in what follows we slightly abuse the notation by using the symbols Π_0, Π_1, Π_2 to refer both to the corresponding problem and to the procedure solving it.

$\Pi_0(q)$**:** For a point $q \in \mathbb{R}^2$, determine whether $q \in K^{\downarrow}$. If $q \in K^{\downarrow}$, the procedure returns a point $\xi \in K$ such that $\xi^{\downarrow} = q$, which serves as a witness to K being nonempty; otherwise (i.e., $q \notin K^{\downarrow}$) it either returns $K = \emptyset$ or a set Ω_q of at most four witness quasi-cylinders in $\mathcal{G}_J$ such that $q \notin \mathcal{I}^{\downarrow}(\Omega_q)$.

$\Pi_1(\ell)$**:** For a line ℓ in $\mathbb{R}^2$ parallel to the y-axis, determine whether $\ell \cap K^{\downarrow} \neq \emptyset$. If $\ell \cap K^{\downarrow} \neq \emptyset$, then the procedure returns a point $\xi \in K$ such that $\xi^{\downarrow} \in \ell$, as a witness to the nonemptiness of K; otherwise it either returns $K = \emptyset$ or a set Ω_ℓ of at most four witness quasi-cylinders in $\mathcal{G}_J$ such that $\mathcal{I}(\Omega_\ell)^{\downarrow} \cap \ell = \emptyset$.

Π_2**:** Determine whether $K^{\downarrow} \neq \emptyset$. If the answer is YES, then return a point $\xi \in K$.

We will later describe the modifications in all these procedures that allows us to return the leftmost point of K if it is not empty.

We answer the query by solving the subproblem Π_2. The algorithm for Π_2 maintains a region $\tau \subset \mathbb{R}^2$ that is guaranteed to contain $K^{\downarrow}$ if $K^{\downarrow} \neq \emptyset$. At each step, it recursively calls Π_1 with some y-parallel lines ℓ, and each of these subprocedures calls in turn Π_0 with some points on the respective line ℓ. If a recursive call for Π_1 or Π_0 either returns $K = \emptyset$ or returns a point $\xi \in K$, the overall algorithm stops right away and returns the same output. In what follows we will not mention this case explicitly and focus on the case when a recursive call returns neither of these outputs. In that case, the recursive call to $\Pi_1(\ell)$ returns a constant-size set $\Omega_\ell \subseteq \mathcal{G}_J$ of witness quasi-cylinders. We compute $Q = \mathcal{I}(\Omega_\ell)$. Since $Q^{\downarrow}$ is convex, $K^{\downarrow} \subseteq Q^{\downarrow}$, and $Q^{\downarrow} \cap \ell = \emptyset$, we can determine the halfplane bounded by ℓ that contains $Q^{\downarrow}$, and thus also $K^{\downarrow}$ (if nonempty), and use this information to shrink the region τ; see below for details. Similarly, as already mentioned, the execution of $\Pi_1(\ell)$ recursively calls $\Pi_0(q)$ for certain points $q \in \ell$. Π_1 uses the witness set Ω_q returned by $\Pi_0(q)$ to shrink the portion of ℓ, which is a progressively shrinking segment (possibly becoming empty eventually), that is guaranteed to contain $\ell \cap K^{\downarrow}$ (if this intersection is nonempty). Finally, $\Pi_0(q)$ is solved by querying M_j, for every $j \in J$, with q. We now provide more details for each of these procedures.

Solving $\Pi_0(q)$. For every $j \in J$, we compute, in $O(\log n)$ time, $\varrho_j(q)$ and $\Omega_j(q)$. If $\varrho_j(q)$ is undefined for some j, i.e., $q \notin K_j^{\downarrow}$, then $q \notin K^{\downarrow}$ and we return NO along with a witness set $\Omega_j(q)$ of at most four upper and lower quasi-cylinders, for which $\mathcal{I}^{\downarrow}(\Omega_j(q))$ does not contain q, as the output witness set Ω_q. So assume that $\varrho_j(q)$ is defined for every $j \in J$. We compute the union $\mathcal{G}_{|q} \subset \mathcal{G}_J$ of the "positive" witness sets returned by each computation (clearly $|\mathcal{G}_{|q}| = O(t)$), and its common intersection $K_{|q}$. If $K_{|q} = \emptyset$, then $K \subseteq K_{|q}$ is also empty and we return $K = \emptyset$. So assume that $K_{|q} \neq \emptyset$. By (C1), $q \in K_{|q}^{\downarrow}$ if and only if $q \in K^{\downarrow}$. Hence, if $q \in K_{|q}^{\downarrow}$, we return any point of $K_{|q} \cap q^{\uparrow}$ (where $q^{\uparrow}$ is the vertical line passing through q) as a witness point of K; otherwise, using (C2), we return a witness set $\Omega_q \subseteq \mathcal{G}_{|q}$ of at most four upper and lower quasi-cylinders, such that $q \notin \mathcal{I}^{\downarrow}(\Omega_q)$.

For each $j \in J$, computing $\varrho_j(q)$ and $\Omega_j(q)$ takes $O(\log n)$ time. By (C1) and (C2), we spend $O(t \log n)$ time to compute $K_{|q}$ (including the test whether $q \in K_{|q}^{\downarrow}$) and Ω_q. Hence, the total time spent by Π_0 is $O(t \log n)$.

Solving $\Pi_1(\ell)$. For a line ℓ in $\mathbb{R}^2$ parallel to the y-axis, we wish to determine whether $K^{\downarrow} \cap \ell \neq \emptyset$, and to report an appropriate witness set when the intersection is nonempty. Let $\Phi = \langle \varphi_1, \ldots, \varphi_u \rangle$ be the sequence of intersection points of the edges of E (the edges of the maps M_j, for $j \in J$) with ℓ, sorted in increasing order of their y-coordinates. We perform a randomized binary search on Φ without computing this sequence explicitly. At each step we have a y-interval $I = (\varphi_a, \varphi_b) \subseteq \ell$ such that $K^{\downarrow} \cap \ell = K^{\downarrow} \cap I$; initially, $I = \ell$. We maintain the invariant that $\Pi_0(\varphi_a), \Pi_0(\varphi_b)$ have already been executed and that $\mathcal{I}^{\downarrow}(\Omega_{\varphi_a} \cup \Omega_{\varphi_b}) \cap \ell \subseteq I$, which implies that $K^{\downarrow} \cap \ell = K^{\downarrow} \cap I$, as desired.

If $a < b - 1$, then we choose a random point φ_s from $I \cap \Phi$ uniformly, using a sequence of operations of type (A2), as described below, and invoke $\Pi_0(\varphi_s)$. If $\Pi_0(\varphi_s)$ returns $K = \emptyset$ or a point $\xi \in K$, Π_1 returns the same output, and stops the overall algorithm. Otherwise, $\Pi_0(q)$ returns NO with a set $\Omega_{\varphi_s} \subseteq \mathcal{G}_J$ of at most four quasi-cylinders such that $\varphi_s \notin \psi := \mathcal{I}^{\downarrow}(\Omega_{\varphi_s})$. If $\psi \cap \ell = \emptyset$, Π_1 returns NO with Ω_{φ_s} as the output witness set Ω_ℓ. If $\psi \cap I = \emptyset$, then $K^{\downarrow} \cap \ell \subseteq \mathcal{I}^{\downarrow}(\Omega_{\varphi_a} \cup \Omega_{\varphi_s} \cup \Omega_{\varphi_b}) \cap \ell = \emptyset$, so, using (C2), we return NO along with a witness set $\Omega_\ell \subseteq \Omega_{\varphi_a} \cup \Omega_{\varphi_s} \cup \Omega_{\varphi_b}$ of at most four quasi-cylinders. Finally, if $\psi \cap I \neq \emptyset$ and lies below (resp., above) φ_s, we shrink I to (φ_a, φ_s) (resp., to (φ_s, φ_b)), and continue the search.

The binary search terminates either when it manages to show that $K = \emptyset$ or finds a point in K (and then the entire algorithm stops), or else when $a = b - 1$. In this case, I lies inside a single (2-dimensional) face f_j of M_j for every $j \in J$. We compute $\mathcal{G}_{|I}$. If $K_{|I} = \emptyset$, we return $K = \emptyset$. By (C1), $K_{|I}^{\downarrow} \cap I = K^{\downarrow} \cap I$, so if $K_{|I}^{\downarrow} \cap I \neq \emptyset$, we return YES with a point $\xi \in K_{|I}$ such that $\xi^{\downarrow} \in I$ (by (C1), $\xi \in K$). Finally, if $K_{|I}^{\downarrow} \cap I = \emptyset$, we conclude that $K^{\downarrow} \cap \ell = \emptyset$ and, using (C2), return a witness subset Ω_ℓ of $\mathcal{G}_{|I}$ of at most four quasi-cylinders for which $\mathcal{I}^{\downarrow}(\Omega_\ell) \cap \ell = \emptyset$. This determines a unique side of ℓ where $K^{\downarrow}$ can lie.

We proceed to describe how to choose a random point φ_s from $I \cap \Phi$. For each $j \in J$, we count the number μ_j of edges of M_j that intersect I. We choose a random integer $j^* \in J$ where $\Pr[j \text{ is chosen}] = \mu_j / \sum_{k \in J} \mu_k$. Next, we choose a random edge γ of M_{j^*} that intersects the interval I, and set $\varphi_s = \gamma \cap \ell$. All this is accomplished by counting and sampling operations of type (A2). By the properties of these operations, we spend $O(\log n)$ time to count the number of edges of M_j intersecting I or to choose a random edge of M_j that intersects I. Hence, φ_s can be chosen in $O(t \log n)$ time. Since the execution of $\Pi_0(\varphi_s)$ also takes $O(t \log n)$ time, each step of the randomized binary search takes $O(t \log n)$ time. The binary search terminates after $O(\log n)$ expected number of steps and computing and manipulating each of the intersections $K_{|I}$ arising during the search takes $O(t \log n)$ time, so the expected running time of $\Pi_1(\ell)$ is $O(t \log^2 n)$.

Solving Π_2. We now describe the procedure Π_2 that detects whether $K^{\downarrow} \neq \emptyset$ (that is, whether $K \neq \emptyset$). We first perform a binary search on $X = \langle x_1, \ldots, x_u \rangle$, the sequence of the x-coordinates of the vertices of all the planar maps M_j for $j = 1, \ldots, 2m$ (recall that X has been constructed explicitly during preprocessing), to find either a y-parallel line $x = x_i$ that intersects $K^{\downarrow}$ or an interval (x_i, x_{i+1}) such that $K^{\downarrow} \subset (x_i, x_{i+1}) \times \mathbb{R}$. At each recursive step, we have an x-interval $I = (x_l, x_r)$ such that $K^{\downarrow}$ (if nonempty) lies in the slab $\sigma := I \times \mathbb{R}$. Initially, $I = (x_1, x_u)$. If $l < r - 1$, we choose $s = \lceil (l + r)/2 \rceil$ and execute $\Pi_1(\ell_s)$, where ℓ_s is the line $x = x_s$. If $\Pi_1(\ell_s)$ returns NO with a witness set Ω_{ℓ_s}, we compute $Q = \mathcal{I}(\Omega_{\ell_s})$. If $Q^{\downarrow} \cap \sigma = \emptyset$, then $K^{\downarrow} = \emptyset$ and we return $K = \emptyset$. Otherwise, if $Q^{\downarrow}$ lies to the right (resp., left) of ℓ_s, we shrink I to (x_s, x_r) (resp., to (x_l, x_s)) and continue with the binary search.

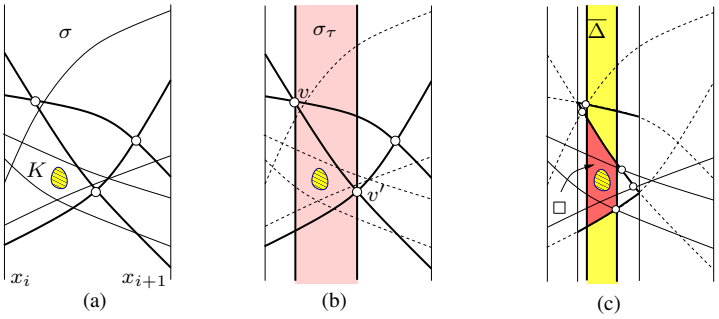


Figure 6. (a) Slab σ and the arcs in E; thick arcs denote those in R. (b) The arrangement $\mathcal{A}(R)$ and the slab σ_τ defined by two consecutive vertices of $\mathcal{A}(R)$. (c) Slab $\overline{\Delta}$ and the new trapezoid $\square$.

The binary search terminates with two consecutive points $x_i, x_{i+1} \in X$ such that $K^{\downarrow} \subset \sigma := (x_i, x_{i+1}) \times \mathbb{R}$ (if nonempty). Since σ does not contain a vertex of any planar map M_j, any y-parallel line inside σ intersects the same set of edges of E (though not necessarily in a fixed order); see Figure 6 (a). We now perform a two-dimensional prune and search to determine whether $K^{\downarrow} \neq \emptyset$.

At each step of the search we have a *pseudo-trapezoid* $\tau \subseteq \sigma$ bounded to the left and right by y-parallel segments and on the top and bottom by (portions of) edges of E, so that *the top and bottom edges of τ do not intersect any arc of E*. Initially, $\tau = \sigma$ (and the top and bottom edges are at infinity).

Let $E_\tau \subset E$ denote the set of arcs that intersect τ. First, assume that $E_\tau \neq \emptyset$. Since the top and bottom edges of τ do not intersect any arc of E and the arcs of E are x-monotone, each arc of E_τ intersects the left (and the right) edge of τ. We fix a sufficiently large constant c. If $|E_\tau| \leq c$, we set $R = E_\tau$; otherwise, we set R to be a random subset of c arcs of E_τ. Using the method described in the Π_1-procedure, R can be chosen in $O(t \log n)$ time. We compute the arrangement $\mathcal{A}(R)$ within τ, and let V_τ denote the sequence of the x-coordinates of all the vertices of $\mathcal{A}(R)$, sorted from left to right. Running a binary search over $v \in V_\tau$, and calling Π_1 at each of the lines $x = v$ of the search, we can determine that $K = \emptyset$, find a point in K, or find, as above, two consecutive values $v, v' \in V_\tau$ such that $K^{\downarrow}$ (if nonempty) lies inside the slab $\sigma_\tau := (v, v') \times \mathbb{R}$; see Figure 6(b).

Since σ_τ does not contain any vertex of $\mathcal{A}(R)$, all y-parallel lines in σ_τ intersect the same subset of edges of $\mathcal{A}(R)$ and in the same order. Let $R' = \langle g_1, \ldots, g_c \rangle$ be the sequence of these arcs sorted from bottom to top in the y-direction. We repeat the following step for each arc $g \in R'$. Let $\Xi_g = \langle \chi_1, \ldots, \chi_t \rangle$ be the sequence of x-coordinates of the intersection points of g with the edges of E, sorted in left to right order. Since g and the other edges are x-monotone elliptic arcs, g can intersect any other edge in at most four points. Without computing Ξ explicitly, we perform a randomized binary search on this sequence, as follows. At each step of the search we have an interval (χ_l, χ_r) of the x-axis delimited by the x-coordinates of two intersection points on g. Using operations of type (A3), we can obtain a random edge of E that intersects the portion of γ between χ_l and χ_r; this is done similarly to the random sampling used in the Π_1-procedure.

The sampled edge can cross γ in up to four points, and we return a random point among them. Note that this is not uniform sampling from Ξ_g, but it deviates from the uniform sampling by a factor of at most 3 and at least $1/3$; that is the probability of a point of Ξ_g to be sampled is at least $1/3$ and at most 3 times its uniform probability. Hence we can choose, in $O(t \log n)$ time, a random value $\chi_s \in (\chi_l, \chi_r) \cap \Xi_g$ and execute $\Pi_1(\ell_s)$ where ℓ_s is the line $x = \chi_s$. Using the witness set Ω_ℓ returned by Π_1, we determine whether $K^\downarrow$ lies to the left or to the right of ℓ_s and shrink the interval to (χ_l, χ_s) or (χ_s, χ_r), accordingly. At the end of the binary search, we have an interval $\Delta_g = (\chi_i, \chi_{i+1})$ such that $K^\downarrow$ lies in the slab $\overline{\Delta}_g := \Delta_g \times \mathbb{R}$. Let $\tilde{g} = g \cap \overline{\Delta}_g$; $\tilde{g}$ lies in a single face f_j of M_j for every $j \in J$. If $K_{|\tilde{g}} = \emptyset$, we return $K = \emptyset$. By (C1), $K^\downarrow_{|\tilde{g}} \cap \tilde{g} = K^\downarrow \cap \tilde{g}$. If $K^\downarrow_{|\tilde{g}} \cap \tilde{g} \neq \emptyset$, we return a point $\xi \in K_{|\tilde{g}}$ such that $\xi^\downarrow \in \tilde{g}$ (again, this is doable by (C1)). If $K^\downarrow_{|\tilde{g}}$ lies below (resp., above) $\tilde{g}$, then we conclude that $K^\downarrow$ (if nonempty) lies below (resp., above) $\tilde{g}$. See Figure 6 (c). Each step of the binary search invokes Π_1 and thus takes $O(t \log^2 n)$ time, so the expected time spent in performing the binary search on g is $O(t \log^3 n)$.

After repeating the above algorithm for every arc in R', we obtain the slab $\overline{\Delta} = \bigcap_{g \in R'} \overline{\Delta}_g$ that must contain $K^\downarrow$, and two arcs $g, g' \in R'$ that are consecutive in the y-order within $\overline{\Delta}$ and $K^\downarrow$ lies between them. (It is also possible that we find two arcs g and g' in R' such that g lies above g' but the algorithm concludes that $K^\downarrow$ (if nonempty) lies above g and below g'. In this case we conclude that $K = \emptyset$ and stop right away.) Let $\square$ be the portion of $\overline{\Delta}$ lying between g_i and g_{i+1}; $\square$ is a pseudo-trapezoid and $K^\downarrow \subseteq \square$. Since $\partial\overline{\Delta}_g$ passes through two consecutive points of Ξ_g, for each $g \in R$, no arc of E intersects g_i or g_{i+1} inside $\overline{\Delta}$. Hence, the top and bottom edges of $\square$ do not intersect any arc of E. See Figure 6 (c).

Let $E_\square \subseteq E$ be the set of arcs that intersect $\square$. Since the interior of $\square$ does not intersect any arc of R, we can ensure, with probability at least $3/4$, that $|E_\square| \leq |E_\tau|/2$, if we choose c sufficiently large. If $|E_\square| > |E_\tau|/2$, we choose a new subset R and repeat the whole step. The expected number of repetitions of the step is a (small) constant. The expected time spent in computing $\square$ after having found the slab σ_τ is $O(t \log^3 n)$. If $|E_\square| \leq |E_\tau|/2$, then we recursively search in $\square$.

The recursion terminates when $E_\tau = \emptyset$. Now τ lies inside a single face f_j of M_j for all $j \in J$. Since $K^\downarrow \subseteq \tau$, (C1) implies that $K^\downarrow \neq \emptyset$ if and only if $K^\downarrow_{|\tau} \cap \tau \neq \emptyset$. By computing $K^\downarrow_{|\tau}$ in $O(t \log n)$ time, we can determine whether $K \neq \emptyset$ (as implied by (C1)). The expected number of steps taken by the two-dimensional prune and search is $O(\log n)$, so the expected running time of Π_2 is $O(t \log^4 n)$. Putting everything together we obtain the following main result of this section.

Theorem 4.1. *Let $\mathcal{B}$ be a set of n congruent balls in $\mathbb{R}^3$, and let $\mathbb{B} = \{\mathcal{B}_1, \ldots, \mathcal{B}_m\}$ be a collection of m subsets $\mathcal{B}_i \subseteq \mathcal{B}$ for $i = 1, \ldots, m$. Then $\mathbb{B}$ can be preprocessed in $O\big(\sum_i |\mathcal{B}_i| \log n\big)$ time into a data structure so that for any $J \subseteq \{1, \ldots, m\}$, one can detect, in $O(|J| \log^4 n)$ randomized expected time, whether $\bigcap_{j \in J} \mathcal{I}(\mathcal{B}_j) \neq \emptyset$ and, if so, return a point in the intersection.*

Returning the leftmost point in the intersection. The above emptiness detection procedure returns an arbitrary point in $\bigcap_{j \in J} \mathcal{I}(\mathcal{B}_j)$. We can modify this procedure to return the leftmost point in $\bigcap_{j \in J} \mathcal{I}(\mathcal{B}_j)$ within the same asymptotic time bound. The above procedure repeatedly shrinks a region τ (a slab or a pseudo-trapezoid), which is bounded from its sides by a pair of y-parallel lines $x = x_l$ and $x = x_r$. To modify the procedure, instead of returning a point $\xi = (x_\xi, y_\xi)$ in $\bigcap_{j \in J} \mathcal{I}(\mathcal{B}_j)$ when such a point is found, we consider the witness set Ω_ξ of at most four upper and lower quasi-cylinders that witness the fact that $\bigcap_{j \in J} \mathcal{I}(\mathcal{B}_j)$ is nonempty. If the leftmost point q of $\mathcal{I}(\Omega_\xi)$ projects to ξ, we conclude that q is the leftmost point of $\bigcap_{j \in J} \mathcal{I}(\mathcal{B}_j)$ and we return q as before. Otherwise, we shrink τ by setting its right boundary to $x = x_\xi$. This modification applies in all relevant steps of the algorithm, including the one that performs a binary search over the arcs of the sample R'. At the end of the algorithm, if we have not yet encountered the leftmost point, τ is fully contained in a single face f_j of M_j for each $j \in J$. We compute $K_{|\tau}$ and return the leftmost point of it.

A slight improvement in the running time. In our application $|J| = O(\log n)$, so the emptiness detection procedure provided by Theorem 4.1 takes $O(\log^5 n)$ time. We can slightly improve this bound, to $O(\log^4 n \log\log n)$, as follows. We first observe that finding the leftmost point in $\bigcap_{j\in J} \mathcal{I}(\mathcal{B}_j)$ is an LP-type problem of constant combinatorial dimension (see [18]), where the constraints are the intersections $\mathcal{I}(\mathcal{B}_j)$, and the value of a subset of J is the (x-coordinate of the) leftmost point in the corresponding intersection. We note that this setup is somewhat nonstandard, because the individual constraints do not have constant complexity, so the primitive operations of violation tests and basis recomputations, which are the building blocks of algorithms for LP-type problems, require some care in their implementation. We can therefore solve this problem by following the algorithm of Gärtner and Welzl [18]. This algorithm relies on two randomized algorithms by Clarkson [12], and the subexponential algorithms of Kalai [22] or Matoušek *et al.* [24]. The first algorithm of Clarkson performs $O(|J|)$ violation tests, each testing whether some point q lies outside some intersection $\mathcal{I}(\mathcal{B}_j)$. Each violation test can be implemented in $O(\log n)$ time, by performing a point location query in M_j (a substep of the procedure $\Pi_0(q)$). This algorithm also makes a constant number of calls to the second algorithm of Clarkson, each time with some set H of $O(\sqrt{|J|})$ constraints as input. The second algorithm, in turn, performs $O(|H|\log|H|)$ violation tests, each implemented in $O(\log n)$ time as above, and makes $O(\log|H|)$ calls to the subexponential algorithm with a set of $O(1)$ constraints as input. We replace this call by a call to our modified emptiness detection procedure from the previous paragraph, which works in $O(\log^4 n)$ randomized expected time due to the constant number of constraints. Altogether, the running time of the new algorithm is $O(|J|\log n + \sqrt{|J|}\log|J|\log n + \log|J|\log^4 n)$, which, for $|J| = O(\log n)$, is $O(\log^4 n \log\log n)$. We thus obtain the final result of this section.

Corollary 4.2. *Let $\mathcal{B}$ be a set of n congruent balls in $\mathbb{R}^3$, and let $\mathbb{B} = \{\mathcal{B}_1, \ldots, \mathcal{B}_m\}$ be a collection of m subsets $\mathcal{B}_i \subseteq \mathcal{B}$ for $i = 1, \ldots, m$. Then $\mathbb{B}$ can be preprocessed in $O\big(\sum_i |\mathcal{B}_i| \log n\big)$ time into a data structure so that for any $J \subseteq \{1, \ldots, m\}$, with $|J| = O(\log n)$, one can detect, in $O(\log|J|\log^4 n) = O(\log^4 n \log\log n)$ randomized expected time, whether $\bigcap_{j\in J} \mathcal{I}(\mathcal{B}_j) \neq \emptyset$ and, if so, return a point in the intersection.*

5 Conclusion

We presented an algorithm for computing the 2-center of a set of points in $\mathbb{R}^3$ that takes near-quadratic time, provided the two centers are not too close to each other. An obvious open problem is to design an algorithm for the 2-center problem that runs in near-quadratic time on all point sets in $\mathbb{R}^3$. Another interesting question is whether the 2-center problem in $\mathbb{R}^3$ is 3SUM-*hard* (see [17] for details), which would suggest that a near-quadratic algorithm is (almost) the best possible for this problem.

Acknowledgment. We would like to thank the anonymous referees for their helpful comments on this paper. In particular, we thank one of the referees for suggesting the improvement of the algorithm in Section 4, as given in Corollary 4.2.

References

[1] P. K. Agarwal, B. Aronov, M. Sharir, and S. Suri, Selecting distances in the plane, *Algorithmica* 9 (1993), 495-514.

[2] P. K. Agarwal, A. Efrat and M. Sharir, Vertical decomposition of shallow levels in 3-dimensional arrangements and its applications, *SIAM J. Comput.* 29 (2000), 912–953.

[3] P. K. Agarwal and C. M. Procopiuc, Exact and approximation algorithms for clustering, *Algorithmica* 33 (2002), 201–226.

[4] P. K. Agarwal and M. Sharir, Planar geometric location problems, *Algorithmica* 11 (1994), 185–195.

[5] M. Bădoiu, S. Har-Peled and P. Indyk, Approximate clustering via core-sets, *Proc. 34th ACM Sympos. Theory Comput.* (2002), 250–257.

[6] R. Brandenberg and L. Roth, New algorithms for k-center and extensions, *J. Combinatorial Optimization* 18 (2009) 376–392.

[7] T. M. Chan, An optimal randomized algorithm for maximum Tukey depth, *Proc. 15th ACM-SIAM Sympos. Discrete Algorithms* (2004), 430–436.

[8] T. M. Chan, Geometric applications of a randomized optimization technique, *Discrete Comput. Geom.* 22 (1999), 547–567.

[9] T. M. Chan, More planar two-center algorithms, *Comput. Geom. Theory Appl.* 13 (1999), 189–198.

[10] B. Chazelle, Cutting hyperplanes for divide-and-conquer, *Discrete Comput. Geom.* 9 (1993), 145–158.

[11] B. Chazelle and J. Matoušek, On linear-time deterministic algorithms for optimization problems in fixed dimension, *J. Algorithms* 21 (1996), 579–597.

[12] K. L. Clarkson, Las Vegas algorithms for linear and integer programming when the dimension is Small, *J. ACM* 42(2) (1995), 488–499.

[13] K. L. Clarkson and P. W. Shor, Applications of random sampling in computational geometry, II, *Discrete Comput. Geom.* 4 (1989), 387–421.

[14] M. de Berg, M. van Kreveld, M. Overmars and O. Schwarzkopf, *Computational Geometry: Algorithms and Applications*, Springer-Verlag, 1997.

[15] H. Edelsbrunner and M. H. Overmars, Batched dynamic solutions to decomposable searching problems, *J. Algorithms* 6 (1985), 515–542.

[16] D. Eppstein, Faster construction of planar two-centers, *Proc. 8th ACM-SIAM Sympos. Discrete Algorithms* (1997), 131–138.

[17] A. Gajentaan and M. H. Overmars, On a class of $O(n^2)$ problems in computational geometry, *Comput. Geom. Theory Appl.* 5(3) (1995), 165–185.

[18] B. Gärtner and E. Welzl, Linear programming - randomization and abstract frameworks, *Proc. 13th Annu. Sympos. Theoret. Aspects Comput. Sci.* (1996), 669–687.

[19] T. Gonzalez, Clustering to minimize the maximum intercluster distance, *Theoret. Comput. Sci.* 38 (1985), 293–306.

[20] J. Hershberger and S. Suri, Finding tailored partitions, *J. Algorithms* 12 (1991), 431–463.

[21] J. Jaromczyk and M. Kowaluk, An efficient algorithm for the Euclidean two-center problem, *Proc. 10th ACM Sympos. Comput. Geom.* (1994), 303–311.

[22] G. Kalai, A subexponential randomized simplex algorithm, *Proc. 24th ACM Sympos. Theory Comput.* (1992), 475–482.

[23] J. Matoušek, Linear optimization queries, *J. Algorithms* 14 (3) (1993), 432–448.

[24] J. Matoušek, M. Sharir, and E. Welzl, A subexponential bound for linear programming, *Algorithmica*, 16 (1996), 498–516.

[25] N. Megiddo and K. Supowit, On the complexity of some common geometric location problems, *SIAM J. Comput.* 13 (1984), 1182–1196.

[26] N. Megiddo, On the complexity of some geometric problems in unbounded dimension, *J. Symbolic Comput.* 10 (1990), 327–334.

[27] N. Sarnak and R. E. Tarjan, Planar point location using persistent search trees, *Comm. ACM* 29 (7) (1986), 669–679.

[28] M. Sharir, A near linear algorithm for the planar 2-center problem, *Discrete Comput. Geom.* 18 (1997), 125–134.