Approximating Smallest Enclosing Balls

Frank Nielsen[1] and Richard Nock[2]

[1] Sony CS Laboratories Inc., Tokyo, Japan
Frank.Nielsen@acm.org
[2] UAG-DSI-GRIMAAG, Martinique, France
Richard.Nock@martinique.univ-ag.fr

Abstract. We present two novel tailored algorithms for computing arbitrary fine approximations of the smallest enclosing ball of balls. The deterministic heuristics are based on solving relaxed decision problems using a primal-dual method.

1 Introduction

The smallest enclosing disk problem dates back to 1857 when J. J. Sylvester [20] first asked for the smallest radius disk enclosing n points on the plane. More formally, let $\mathrm{Ball}(P,r)$ denote the ball of center P and radius r: $\mathrm{Ball}(P,r) = \{X \in \mathbb{E}^d \mid ||PX|| \leq r\}$, where $||\cdot||$ denotes the L_2-norm of Euclidean space $\mathbb{E}^d$. Let $\mathcal{B} = \{B_1, ..., B_n\}$ be a set of n d-dimensional balls, such that $B_i = \mathrm{Ball}(P_i, r_i)$ for $i \in \{1, ..., n\}$. Denote by $\mathcal{P}$ the ball centers $\mathcal{P} = \{P_1, ..., P_n\}$. The smallest enclosing ball of $\mathcal{B}$ is the unique ball [22], $B^* = \mathrm{SEB}(\mathcal{B}) = \mathrm{Ball}(C^*, r^*)$, fully enclosing $\mathcal{B}$ ($\mathcal{B} \subseteq \mathrm{Ball}(C^*, r^*)$) of minimum radius r^*. Given a ball B, denote by $r(B)$ its radius and $C(B)$ its center. Let $x_i(P)$ denote the i-th coordinate of point P ($1 \leq i \leq d$). The smallest enclosing ball problem is also refered in the literature as the minimum enclosing ball, minimum spanning ball, minimum covering sphere, Euclidean 1-center, d-outer radius, minimum bounding sphere, or minimax problem in facility locations, etc. The smallest enclosing ball, as a fundamental primitive, finds many applications in computer graphics (collision detection, visibility culling, ...), machine learning (support vector clustering, similarity search, ...), metrology (roundness measurements, ...), facility locations (base station locations, ...), and so on. Notice that in the aforementioned applications, approximate solutions is often enough.

We survey below the main algorithms for computing the exact or approximate smallest enclosing balls. We classify previous work in Section 2 according to three algorithmic paradigms: (1) combinatorial algorithms, (2) numerical algorithms and (3) hybrid algorithms. Section 3 describes a general filtering mechanism for computing the maximum distance set-element that is then used in Section 4 to improve an implementation of a recent core-set approximation algorithm [3]. Section 5 presents a novel core-set primal-dual tailored method based on solving relaxed decision problems. Section 6 gives an alternative approach better suited for small dimensions and discusses on the algebraic degree of predicates.

2 Previous Work

Combinatorial. The smallest enclosing ball complexity was only settled in 1984 by N. Megiddo's first linear-time prune-and-search algorithm [16] for solving linear programs in fixed dimension. Later, the method was extended to the case of balls [17]. Since the smallest enclosing ball is unique and defined by at most $d+1$ support points (or balls) in strictly convex position (implying being affinely independent as well), a brute-force combinatorial algorithm requires $O_d(n^{d+2})$ time (linear memory). A major breakthrough was obtained by E. Welzl [22] who describes an elegant randomized almost tight expected $\lfloor (e-1)(d+1)! \rfloor n$ time[3] algorithm. The number of basis computations is shown to be $\tilde{O}(\log^j n)$ $(2 \leq j \leq d+1)$, so that most of the time of the algorithm is spent by checking that points/balls are inside some candidate ball.[4] For point sets being vertices of a regular simplex, the algorithm exhibits the *curse of dimensionality* as it requires $\Omega(2^d)$ recursive calls, thus limiting its tractibility up to a few dozen dimensions in practice. Recently, Chernoff-type tail bound has been given for nondegenerate input by B. Gärtner and E. Welzl [10]. Although it gives a better understanding of the power of randomization, tight worst-case bound is unknown[5] as is also the tail estimate in case of cospherical point sets. Subexponential running time was obtained by B. Gärtner [7] who described a general randomized algorithm for the class of so-called abstract optimization problems (AOP). Focusing on small instances (*i.e.*, $n = O(d)$), B. Gärtner and E. Welzl [11] presents a practical randomized approach for affinely independent points using $\tilde{O}(1.5^n)$ basis computations. T. Szabo and E. Welzl [21] further improve the bound to $\tilde{O}(1.47^n)$ using the framework of unique sink orientations of hypercubes. So far, B. Chazelle and J. Matoušek gave the current best $O(d^{O(d)}n)$ deterministic time algorithm [4]. From the pratical viewpoint, B. Gärtner [8] updated the move-to-front heuristic of E. Welzl [22] by introducing a pivot mechanism and improving the robustness of basis computations. Furthermore, K. Fischer *et al.* [6] describe a simplex-like pivoting combinatorial algorithm with a Bland-type rule that guarantees termination based on the seminal idea of T. Hopp et C. Reeve [14] of deflating an enclosing sphere: They devise a dynamic data-structure for maintaining intermediate candidate balls and a robust floating-point implementation is tested with point sets up to dimension[6] 10000. Overall complexity is $O(d^3+d^2l)$, where $l \leq \binom{n}{d+1}$ is a finite number of iterations; In practice, although the algorithm requires algebraic degree 2 on the rationals, they observe *good experimental* floating-point errors of at most 10^4 times the machine precision. For ball sets, K. Fischer and B. Gärtner show [5] that for affinely independent ball centers that E. Welzl's algorithm [22] extends to balls

[3] $e \simeq 2.71828182846...$ is the irrational number such that $\log e = 1$.

[4] This may explain why descriptions of computing the primitives were omitted in [22] since $\sum_{i=2}^{d+1}(2+\ln n)^i = o_d(n)$.

[5] That is to know the worst-case geometric configuration that implies a worst number of recursive calls (geometric realization of permutations).

[6] In fact, T. Hopp and C. Reeve [14] reported experimentally a time complexity of $\tilde{O}(d^{2.3}n)$ for uniform spherical data sets.

and provide a linear programming type (LP-type) algorithm which runs in expected $\tilde{O}(2^{O(d)}n)$-time. The combinatorial algorithms described so far compute the exact smallest enclosing ball (*i.e.*, $\epsilon = 0$), report a support point/ball set and look similar to those handling linear programming. Notice that the smallest enclosing ball problem, as LP, is not known to be strongly polynomial (see P. Gritzmann and V. Klee [12] for a weakly polynomial algorithm).

Numerical. Let $d_2(\mathcal{A}, \mathcal{B})$ denote the maximum distance between all pairs (A, B) ($A \in \mathcal{A}$ and $B \in \mathcal{B}$). Observe that picking any point $P \in \mathcal{B}$ gives a 2-approximate ball $\text{Ball}(P, d_2(P, \mathcal{B}))$ (*i.e.*, $\epsilon = 1$). This allows to easily convert from relative to absolute approximation values. Motivated by computer graphics applications, J. Ritter [19] proposes a simple and fast constant approximation of the smallest enclosing ball that can be extended straightforward for points/balls in arbitrary dimension. Tight worst-case approximation ratio is unknown but can be as bad as 18.3 percents.[7] It is quite natural to state the smallest enclosing ball problem as a mathematical program. In facility locations, the smallest enclosing ball is often written as $\min_{C \in \mathbb{E}^d} F_{\mathcal{B}}(C)$ where $F_{\mathcal{B}}(X) = \max_{i \in \{1,\ldots,n\}} d_2(X, \mathcal{B})$. Since the minimum is unique, we obtain the circumcenter as $C^* = \text{argmin}_{C \in \mathbb{E}^d} F_{\mathcal{B}}(C)$. Using the ellipsoid method for solving approximately convex programs (CP), we get a $(1+\epsilon)$-approximation in $O(d^3 n \log \frac{1}{\epsilon})$ time [13]. B. Gärtner and S. Schönherr [9] describes a generic quadratic programming (QP) solver tuned up for dense problems with few variables, as it is the case for solving basic instances. The solver *behaves polynomially* but requires arbitrary-precision linear algebra that limits its use to a few hundred dimensions. Recently, another method which turns out to perform so far best in practice, is the second-order cone programming [24] (SOCP) and requires $O(\sqrt{n} \log \frac{1}{\epsilon})$ iterations [18] using interior-point methods. Each iteration can be performed in $O(d^2(n+d))$ time for the smallest enclosing ball. G. Zhou *et al.* [24] present another algorithm, based on providing a smooth approximation of the nondifferentiable minimax function $F_{\mathcal{B}}(\cdot)$ using so-called log-exponential aggregation functions, that scale well with dn and $\frac{1}{\epsilon}$. For coarse ϵ values, say $\epsilon \in [0.001, 0.01]$, subgradient steepest-descent methods can be used as it first converges fast before slowly zigzagging towards the optimum. These numerical techniques rely on *off-the-shelves* optimization procedures that have benefited from extensive code optimization along the years but seem not particularly tuned up for the specific smallest enclosing ball problem.

Hybrid. An ϵ-core set of $\mathcal{P}$ is a subset $\mathcal{C} \subseteq \mathcal{P}$ such that the smallest enclosing ball of $\mathcal{C}$ expanded by a factor of $1+\epsilon$ fully covers set $\mathcal{P}$. Surprisingly, it was shown by M. Bădoiu *et al.* [2] that for any $\epsilon > 0$ there is a core set of size independent of dimension d. The bound was later improved to the tight $\frac{1}{\epsilon}$ value [3]. Note that since the smallest enclosing ball is defined by at most $d+1$ points/balls,

[7] *E.g.*, considering a regular simplex in dimension 2. In [19], J. Ritter evaluates it to "around" 10 percents. X. Wu. [23] suggests a variant based on finding principal axis as a preprocessing stage of J. Ritter's greedy algorithm. It requires roughly twice more time and do not guarantee to perform better. (Actually, we found it experimentally worse sometimes.)

the result is combinatorically meaningful for $\frac{1}{\epsilon} \leq d+1$. Besides, they also give a simple iterative $O(\frac{dn}{\epsilon^2})$-time algorithm (see procedure *SimpleIterativeBall* below) to compute a $(1+\epsilon)$-approximation of the smallest enclosing ball, for any $\epsilon > 0$. Combining the ellipsoid numerical approximation method with the combinatorial core-set approach yields a $O(\frac{dn}{\epsilon} + \frac{d}{\epsilon^4})$-time hybrid algorithm. P. Kumar *et al.* [15] relies on the work of [24] to obtain a better $O(\frac{dn}{\epsilon} + \frac{1}{\epsilon^{\frac{3}{2}}} \log \frac{1}{\epsilon})$-time bound.[8] S. Har-Peled mentioned an unpublished $O(\frac{dn}{\epsilon} + \frac{1}{\epsilon^2} \log^2 \frac{1}{\epsilon})$-time algorithm, so that the hybrid algorithm runs in $O(\frac{dn}{\epsilon} + \frac{1}{\epsilon^4} \log^2 n)$-time. Although not explicitly stated in the pioneer work of [2], the algorithms/bounds are still valid for ball sets (also noticed by [15]).

Our contributions. Although combinatorial algorithms exist for the smallest enclosing ball of points in very large dimensions ($d \simeq 10000$) that prove efficient in practice but lacks deterministic bound (i.e, tight worst-case analysis), we would like to emphasize on the merits of computing approximate solutions: (i) guaranteed worst-case time dependent on $\frac{1}{\epsilon}$ (the less demanding, the faster), (ii) very short code: no basis computations of at most $d+1$ points/balls are required, (iii) no special care are required for handling degeneracies (*i.e.*, co-spherical points), (iv) stable: use predicates of lower degrees (see Section 6). Our contributions are summarized as follows: (i) We show an effective implementation of approximate enclosing balls of core-sets ($d \simeq 15000$ and $\epsilon \simeq 1\%$) based on distance filtering, (ii) We describe a new tailored core-set algorithm for dual decision problems, (iii) We propose an alternative effective algorithm for small dimensions, (iv) we review algorithm performances according to experiments obtained on a common platform.

3 Distance Point-Set Queries

Often, we need to compute the distance, $d_2(P, \mathcal{B})$, from a query point P to a point/ball set $\mathcal{B}$. A naive algorithm, computing distance pairs iteratively, requires $O(dn)$ time per query so that q farthest queries $d_2(\cdot, \mathcal{B})$ cost overall $O(qdn)$ time. When dimension d is large, say $d \geq 100$, computing distances of query point/set become in itself an expensive operation. Observe that $d_2(X,Y) = ||X-Y|| = \sqrt{\sum_{i=1}^{d}(X_i - Y_i)^2}$ can be written as $||X-Y||^2 = ||X||^2 + ||Y||^2 - 2 < X, Y >$, where $<,>$ denotes the vector dot product: $< X, Y >= \sum_{i=1}^{d} X_i Y_i = X^T Y$. Using Cauchy-Schwarz inequality, we have $| < X, Y > | \leq ||X||\ ||Y||$. Therefore, the distance is upper bounded by $\sqrt{||X||^2 + ||Y||^2 + 2\sqrt{||X||^2||Y||^2}} \geq ||X-Y||$. Thus when answering q farthest queries, we can first build lookup tables of $||P_i||^2$ ($P_i \in \mathcal{B}$) in a preprocessing stage in $O(dn)$ time and then use a simple distance filtering mechanism. That is, when iteratively seeking for the maximum distance given a query point X and set $\mathcal{B}$, we skip in $O(1)$ time evaluating distance $d_2(X, P_i)$ if the so far maximum distance is above the upper bound given by

[8] More precisely, $O(\frac{dn}{\epsilon} + \frac{d^2}{\epsilon^{\frac{3}{2}}}(\frac{1}{\epsilon} + d) \log \frac{1}{\epsilon})$-time.

the Cauchy-Schwarz inequality. For sets drawn from statistical distribution, let $\bar{\alpha}$ be the expected number of skipped distances, we answer q queries in $O(d(n+q)+q(1-\bar{\alpha})dn)$ time. For uniform d-cube distributions or normal distributions we observe experimentally $\bar{\alpha} \stackrel{n}{\rightarrow} 1$ (thus for $n \geq \frac{1}{\epsilon^2}$, the algorithm converges towards optimal linear $O(dn)$ time), for uniform distributions on the d-sphere, we conversely observe $\bar{\alpha} \stackrel{n}{\rightarrow} 0$. This approach extends to ball sets as well but requires extra square-root operations in order to handle ball radii.

4 Approximating Smallest Enclosing Balls of Core-Sets

Although M. Bădoiu and K. Clarkson's algorithm [3] (procedure *SimpleIterativeBall* below) extends to ball sets as well, for ease of description, we consider here point sets. The algorithm looks like *gradient-type*[9], but it is not as we noticed experimentally that the radii sequence of enclosing balls is not necessary decreasing. Given a current circumcenter, the procedure finds a farthest point of $\mathcal{B}$ to walk towards in $O(dn)$ time bypassing the costly $O(d^2n)$ time Jacobian computation required in a steepest-descent optimization. Overall cost is $O(\frac{dn}{\epsilon^2})$ time as we need to perform $\lfloor \frac{1}{\epsilon^2} \rfloor$ iterations. Using this elegant algorithm and coupling it with approximations of smallest enclosing balls of core-sets (see [2]), we obtain a $O(\frac{dn}{\epsilon} + \frac{d}{\epsilon^4})$-time algorithm (procedure *ApproximateCoreSet*). For $\frac{1}{\epsilon} = O(\sqrt[3]{n})$, the bottleneck of the algorithm is finding the core-set rather than the overall cost of simple loops.

```
1  SimpleIterativeBall(B, ε);
2  Pick arbitrary C_1 ∈ S; i ← 1;
3  a = ⌊1/ε²⌋;
4  while i ≤ a do
5      m = argmax_j ||C_i S_j|| /* Distance filtering */;
6      C_{i+1} = C_i + 1/(i+1) (S_m − C_i);
7      i ← i + 1;
8  r_a = d_2(C_a, S);
9  return Ball(C_a, r_a);
10 ApproximateCoreSet(B, ε);
11 γ = ε/3; δ = ε/3 /* Guarantee (1+δ)(1+γ) ≤ 1+ε for any ε ≤ 1 */;
12 C_1 ← {B_1}; r_1 = 0; i ← 1;
13 while d_2(C_i, B) ≥ (1+δ) r_i do
14     k = argmax_i d_2(C_i, B) /* Distance filtering */;
15     C_{i+1} ← C_i ∪ {B_k};
16     K_{i+1} ← SimpleIterativeBall(C_{i+1}, γ);
17     C_{i+1} ← C(K_{i+1}); r_{i+1} ← r(K_{i+1});
18     i ← i + 1;
19 return Ball(C_i, r_i);
```

[9] M. Bădoiu and K. Clarkson used the term gradient-like [3].

Plugging the distance filtering mechanism of Section 3, for uniform distribution of ball sets with $d \simeq 10000, n = d + 1, \epsilon \simeq 0.01$, the algorithm requires a few seconds on current commodity PCs for a mere 30-line C code. It performs better in practice than the steepest-descent method. The algorithm is adaptive according to the core-set size, bounded by $\frac{6}{\epsilon}$, but not in the iteration process of [3] as we need to loop exactly $\lfloor \frac{9}{\epsilon^2} \rfloor$ time.[10] Theoretically, this algorithm is only slightly outperformed by a SOCP solver, but its extreme simplicity coupled with the distance filtering trick make it attractive for machine learning applications.

5 Core-Sets for Decision Problems

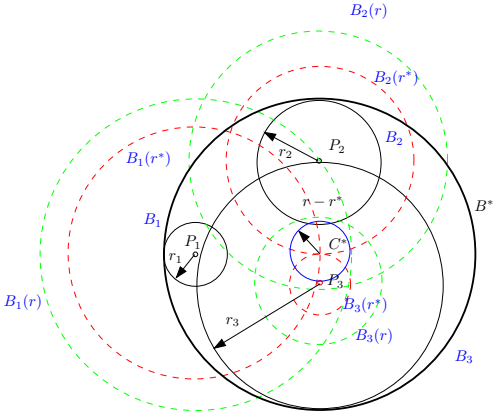

Fig. 1. Covering/piercing duality. Balls B_1, B_2, B_3 are associated to corresponding dashed balls $B_1(r), B_2(r), B_3(r)$ such that $C(B_i(r)) = P_i$ and $r(B_i(r)) = r - r_i$ for $i \in \{1, 2, 3\}$. We have $B_1(r^*) \cap B_2(r^*) \cap B_3(r^*) = \{C^*\}$. For $r \geq r^*$, there exists a ball of radius $r - r^*$ fully contained in $B_1(r) \cap B_2(r) \cap B_3(r)$.

Our novel approximation algorithms proceed by solving dual piercing *decision problems* (see Figure 1): given a set of balls $\mathcal{P} = \{B_i = \text{Ball}(P_i, r_i), i \in \{1, ..., n\}\}$ and some $r \geq 0$, determine whether $\cap\mathcal{B}(r) = \cap_{i \in \{1,...,n\}} B_i(r) = \emptyset$ or not, where $B_i(r) = \text{Ball}(P_i, r - r_i)$. We relax the 1-piercing point problem to that of a common piercing ϵr^*-ball (*i.e.*, a ball of radius ϵr^*): Namely, report whether there exists a ball $B = \text{Ball}(C, \epsilon r^*)$ such that $B \subseteq \cap\mathcal{B}(r)$ or not (see Figure 1).

Lemma. For $r \geq r^*$, there exists a ball B of radius $r(B) = r - r^*$ centered at $C(B) = C^*$ fully contained inside $\cap\mathcal{B}(r)$.

Proof. In order to ensure that C^* is in each $B_i(r)$, a sufficient condition is to have $r \geq \max_i\{r_i + d_2(P_i, C^*)\}$. Since $B_i \subseteq \text{Ball}(C^*, r^*), \forall i \in \{1, 2, ..., n\}$, we have $\max_i\{r_i + d_2(P_i, C^*)\} \leq r^* (\star)$. Thus, provided $r \geq r^*$, we have $C^* \in \cap\mathcal{B}(r)$. Now, notice that $\forall i \in \{1, 2, ..., n\}, \forall 0 \leq r' \leq (r - r_i) - d_2(P_i, C^*), \text{Ball}(C^*, r') \subseteq B_i(r)$. Thus, if we ensure that $r' \leq r - \max_i(r_i + d_2(P_i, C^*))$, then $\text{Ball}(C^*, r') \subseteq \cap\mathcal{B}(r)$. From ineq. $(\star)$, we choose $r' = r - r^*$ and obtain the lemma (see Figure 1). □

The algorithm, detailed in procedure *DecisionProblem* for point sets, builds a core-set (sets $\mathcal{C}_i$'s) iteratively for the decision problem by narrowing the feasible

[10] It is of practical interest to find a better stopping criterion.

domain for circumcenter C^*. It is a primal-dual method since that for solving dual ball piercing problem, it requires to solve primal smallest enclosing balls.

Algorithm: *DecisionProblem*$(\mathcal{B}, \epsilon)$

Let rr be the radius obtained from a trivial 2-approximation algorithm;
Choose arbitrary $P_1 \in \mathcal{P}$; $\mathcal{C}_1 \leftarrow \{P_1\}$; $r_1 \leftarrow 0$; $i \leftarrow 1$;
while $r - r_i \geq \epsilon \frac{rr}{2}$ **do**
    Let $L_i : P_i + \lambda x_d$ /* x_d denote the unit vector of the d-th coordinate axis */;
    $\mathcal{B}_{L_i} = \{B \cap L_i \mid B \in \mathcal{B}\}$;
    if $\cap\mathcal{B}_{L_i} \neq \emptyset$ **then**
        return YES /* $r \geq r^*$ */
    else
        if $\exists B | B \cap L_i = \emptyset$ **then**
            $\mathcal{C}_{i+1} = \mathcal{C}_i \cup \{B\}$;
        else
            Let B_k and B_l such that $(B_k \cap L_i) \cap (B_l \cap L_i) = \emptyset$;
            $\mathcal{C}_{i+1} = \mathcal{C}_i \cup \{B_k, B_l\}$;
    $i \leftarrow i + 1$; $K_i = \text{SEB}(\mathcal{C}_i)$ /* Primal-Dual */ ;
    if $r(K_i) > r$ **then**
        return NO /* $r^* > r$ */
    $P_i = C(K_i)$;
return MAYBE /* $r - r^* \leq \epsilon r^*$ */ ;

Let k denote the maximum number of iterations of the while loop. Observe that balls B already chosen in some core-set $\mathcal{C}_i$ are necessarily pierced by points $C(K_j), j \geq i + 1$. Indeed, since $C(K_i)$ is the center of the smallest enclosing ball of the centerpoints of balls of radius r of $\mathcal{C}_i$, and $r_i = r(K_i) \leq r$, we have $d_2(C(K_i), C(B)) \leq r$ for all $B \in \mathcal{C}_i$. Moreover, since $\cap\mathcal{C}_{i+1} \subset \cap\mathcal{C}_i$ and because the smallest enclosing ball is unique, we have $r_{i+1} > r_i$. Clearly, we have $|\mathcal{C}_i| \leq 2i$. We show that k is a function depending only on d and ϵ, independent of n. Let $v_d(r)$ denote the volume of a d-dimensional ball of radius r. We have $\cap\mathcal{C}_{i+1} \subset \cap\mathcal{C}_i$ for all i. Let K_i be the unique maximal ball contained in $\cap\mathcal{C}_i$ (obtained from the smallest enclosing ball of the centers of balls contained in $\mathcal{C}_i$). If $C(K_i)$, the center of ball K_i, does not fully pierce $\mathcal{B}$, then there exists either one ball M_i or two balls M_i and N_i such that their intersection A_i (either $A_i = M_i$ or $A_i = M_i \cap N_i$) does not contain $C(K_i)$. Since A_i is convex, this means that there exists an hyperplane H_i separating A_i from $C(K_i)$. Let H_i' be an hyperplane parallel to H_i and passing through $C(K_i)$, $H_i'^+$ be the halfspace not containing A_i. Since $\cap\mathcal{C}_{i+1} \subset \cap\mathcal{C}_i$, we have $\text{vol}(\mathcal{C}_{i+1}) \leq \text{vol}(\mathcal{C}_i) - \frac{1}{2} v_d(r(K_i))$. Since $r(K_i) \geq \epsilon r^*$ and $\text{vol}(C_1) \leq v_d(2r^*)$, we get a sloppy upperbound $k = O(\frac{1}{\epsilon})^d$. In a good scenario, where we split in half the volume of $\cap\mathcal{C}_i$, we get $k = O(d \log_2 \frac{1}{\epsilon})$, yielding to an overall $O(d^2 n \log_2 \frac{1}{\epsilon}) + O_{d,\epsilon}(1)$ time algorithm (improve by a factor $O(d)$ over the ellipsoid method). We observe experimentally that k tends indeed to behave as $O_d(\log \frac{1}{\epsilon})$ and that the core-set sizes are similar to the ones obtained by M. Bădoiu and K. Clarkson's algorithm. By solving $O(\log \frac{1}{\epsilon})$ decision problems, we thus obtain a $(1 + \epsilon)$-approximation of the smallest enclosing ball.

6 Small dimensions revisited

In this section, the key difference with the previous heuristic is that dual problem sizes to solve does not depend on ϵ but are exponentially dependent on d.

Solving planar decision problems. Let $[n] = \{1, ..., n\}$ and $[x_m, x_M]$ be an interval on the x-axis where an ϵr^*-disk center might be located if it exists. (That is $x(C) \in [x_m, x_M]$ if it exists.) We initialize x_m, x_M as the x-abscissae extrema: $x_m = \max_{i\in[n]}(x_i) - r,\ x_M = \min_{i\in[n]}(x_i) + r$. If $x_M < x_m$ then clearly vertical line $L : x = \frac{x_m+x_M}{2}$ separates two extremum disks (those whose corresponding centers give rise to x_m and x_M) and therefore $\mathcal{B}(r)$ is not 1-pierceable (therefore not ϵr^*-ball pierceable). Otherwise, the algorithm proceeds by dichotomy. Let $e = \frac{x_m+x_M}{2}$ and let L denotes the vertical line $L : x = e$. Denote by $\mathcal{B}_L = \{B_i \cap L | i \in [n]\}$ the set of n y-intervals obtained as the intersection of the disks of $\mathcal{B}$ with line L. We check whether $\mathcal{B}_L = \{B_i \cap L = [a_i, b_i] | i \in [n]\}$ is 1-pierceable or not. Since $\mathcal{B}_L$ is a set of n y-intervals, we just need to check whether $\min_{i\in[n]} b_i \geq \max_{i\in[n]} a_i$ or not. If $\cap\mathcal{B}_L \neq \emptyset$, then we have found a point $(e, \min_{i\in[n]} b_i)$ in the intersection of all balls of $\mathcal{B}$ and we stop recursing. (In fact we found a $(x = e, y = [y_m = \max_i a_i, y_M = \min_i b_i])$ vertical piercing segment.) Otherwise, we have $\cap\mathcal{B}_L = \emptyset$ and need to choose on which side of L to recurse. W.l.o.g., let B_1 and B_2 denote the two disks whose corresponding y-intervals on L are disjoint. We choose to recurse on the side where $B_1 \cap B_2$ is located (if the intersection is empty then we stop by reporting the two non intersecting balls B_1 and B_2). Otherwise, $B_1 \cap B_2 \neq \emptyset$ and we branch on the side where $x_{B_1B_2} = \frac{x(C(B_1))+x(C(B_2))}{2}$ lies. At each stage of the dichotomic process, we halve the x-axis range where the solution is to be located (if it exists). We stop the recursion as soon as $x_M - x_m < \epsilon\frac{r}{2}$. Indeed, if $x_M - x_m < \epsilon\frac{r}{2}$ then we know that *no center of a ball* of radius ϵr is contained in $\cap\mathcal{B}$. (Indeed if such a ball exists then *both* $\cap\mathcal{B}_{L(x_m)} \neq \emptyset$ and $\cap\mathcal{B}_{L(x_M)} \neq \emptyset$.) Overall, we recurse at most $3 + \lceil\log_2 \frac{1}{\epsilon}\rceil$ times since the initial interval width $x_M - x_m$ is less than $2r^*$ and we consider $r \geq \frac{r^*}{2}$. Thus, by solving $O(\log_2 \frac{1}{\epsilon})$ decision problems (dichotomy search), we obtain a $O(n \log_2^2 \frac{1}{\epsilon})$-time deterministic $(1+\epsilon)$-approximation algorithm. We bootstrap this algorithm in order to get a $O(n \log_2 \frac{1}{\epsilon})$-time algorithm. The key idea is to shrink potential range $[a, b]$ of r^* by selecting iteratively different approximation ratios ϵ_i until we ensure that, at kth stage, $\epsilon_k \leq \epsilon$. Let Ball(C, r) be a $(1+\epsilon)$-approximation enclosing ball. Observe that $|x(C) - x(C^*)| \leq \epsilon r^*$. We update the x-range $[x_m, x_M]$ according to the so far found piercing point abcissae $x(C)$ and current approximation factor. We start by solving the approximation of the smallest enclosing ball for $\epsilon_1 = \frac{1}{2}$. It costs $O(n \log_2 \frac{1}{\epsilon_1}) = O(n)$. Using the final output range $[a, b]$, we now have $b - a \leq \epsilon_1 r^*$. Consider $\epsilon_2 = \frac{\epsilon_1}{2}$ and reiterate until $\epsilon_l \leq \epsilon$. The overall cost of the procedure is $\sum_{i=0}^{\lceil\log_2 \frac{1}{\epsilon}\rceil} O(n \log_2 2) = O(n \log_2 \frac{1}{\epsilon})$. The method extends to disks as well. We report on timings obtained from experiments done on 1000 trials for uniformly distributed 100000-point sets in a unit ring of width 2ϵ ($\bigodot$) or unit square ($\square$). Maximum (max.) and average (avg.) running times are in fractions of a second obtained by a 30-line

C code on an Intel 1.6 GHz processor. (See the public code of D. E. Eberly at `http://www.magic-software.com` for a randomized implementation.)

Method/Distribution	$\square$ Square max	$\odot$ Ring max	$\square$ Square avg	$\odot$ Ring avg
D. E. Eberly ($\epsilon = 10^{-5}$)	0.7056	0.6374	0.1955	0.2767
J. Ritter [19] ($\epsilon > 0.18$)	0.0070	0.0069	0.0049	0.0049
2nd Method ($\epsilon = 10^{-2}$)	0.0343	0.0338	0.0205	0.0286
2nd Method ($\epsilon = 10^{-3}$)	0.0515	0.0444	0.0284	0.0405
2nd Method ($\epsilon = 10^{-5}$)	0.0719	0.0726	0.0473	0.0527

Predicate degree. Predicates are the basic computational atoms of algorithms that are related to their numerical stabilities. D. E. Eberly uses the *InCircle* containment predicate of algebraic degree 4 on integers ($d+2$ in dimension d for integer arithmetic. The degree drops to 2 if we consider rational arithmetic [5]). We show how to replace the predicates of algebraic degree 4 by predicates of degree 2 for integers: "Given a disk center (x_i, y_i) and a radius r_i, determine whether a point (x, y) is inside, on or outside the disk". It boils down to compute the sign of $(x-x_i)^2+(y-y_i)^2-r_i^2$. This can be achieved using another dichotomy search on line $L : x = l$. We need to ensure that if $y_m > y_M$, then there do exist two disjoint disks B_m and B_M. We regularly sample line L such that if $y_m > y_M$, then there exists a sampling point in $[y_M, y_m]$ that does not belong to both disks B_m and B_M. In order to guarantee that setting, we need to ensure some *fatness* of the intersection of $\cap\mathcal{B}(r)\cap L$ by recursing on the x-axis until we have $x_M - x_m \leq \frac{\epsilon}{\sqrt{2}}$. In that case, we know that if there was a common ϵr^*-ball intersection, then its center x-coordinate is inside $[x_m, x_M]$: this means that on L, the width of the intersection is at least $\frac{\epsilon}{\sqrt{2}}$. Therefore, a regular sampling on vertical line L with step width $\frac{\epsilon}{\sqrt{2}}$ guarantees to find a common piercing point if it exists. A straightforward implementation would yield a time complexity $O(\frac{n}{\epsilon}\log_2\frac{1}{\epsilon})$. However, it is sufficient for each of the n disks, to find the upper most and bottom most lattice point in $O(\log_2\frac{1}{\epsilon})$-time using the floor function. Using the bootstrapping method, we obtain a $O(n\log_2\frac{1}{\epsilon})$ time using integer arithmetic with algebraic predicates InCircle of degree 2. In dimension 3 and higher, the dimension reduction algorithm extends with a running time $O_d(n\log_2\frac{1}{\epsilon})$. As a side-effect, we improve the result of D. Avis and M. Houle [1] for the following problem: Given a set $\mathcal{B}$ of n d-dimensional balls of $\mathbb{E}^d$, we can find whether $\cap\mathcal{B} = \emptyset$ or report a common intersection point in $\cap\mathcal{B}$ in deterministic $O_d(n^d \log n)$ time and $O_d(n^d)$ space.

References

1. Avis D, Houle ME (1995) Computational aspects of Helly's theorem and its relatives. Int J Comp Geom Appl 5:357-367

2. Bădoiu M, Har-Peled S, Indyk P (2002) Approximate clustering via core-sets. Proc 34th IEEE Sympos Found Comput Sci (FOCS), pp 250-257. DOI 10.1145/509907.509947
3. Bădoiu M, Clarkson K (2003) Optimal core-sets for balls. Proc 14th ACM-SIAM Sympos Discrete Algorithms (SODA), pp 801-802
4. Chazelle B, Matoušek J (1996) On linear-time deterministic algorithms for optimization problems in fixed dimension. J Algorithms 21:579-597. DOI 10.1006/jagm.1996.0060
5. Fischer K, Gärtner B (2003) The smallest enclosing ball of balls: combinatorial structure and algorithms. Proc 19th ACM Sympos Comput Geom (SoCG), pp 292-301. DOI 10.1145/777792.777836
6. Fischer K, Gärtner B, Kutz M (2003) Fast smallest-enclosing-ball computation in high dimensions. Proc 11th Annu European Sympos Algorithms (ESA), LNCS 2832:630-641
7. Gärtner B (1995) A subexponential algorithm for abstract optimization problems. SIAM J Comput 24:1018-1035. DOI 10.1137/S0097539793250287
8. Gärtner B (1999) Fast and robust smallest enclosing balls. Proc 7th Annu European Sympos Algorithms (ESA), LNCS 1643:325-338
9. Gärtner B, Schönherr S (2000) An efficient, exact, and generic quadratic programming solver for geometric optimization. Proc 16th ACM Sympos Comput Geom (SoCG), pp 110-118. DOI 10.1145/336154.336191
10. Gärtner B, Welzl E (2000) On a simple sampling lemma. Electronic Notes Theor Comput Sci (eTCS), vol 31
11. Gärtner B, Welzl E (2001) Explicit and implicit enforcing: randomized optimization, Computational Discrete Mathematics (Advanced Lectures), LNCS 2122:25-46
12. Gritzmann P, Klee V (1993) Computational complexity of inner and outer j-radii of polytopes in finite-dimensional normed spaces. Mathemat Program. 59(2):163-213
13. Grötschel M, Lovasz L, Schrijver A (1993) Geometric algorithms and combinatorial optimization. Springer-Verlag
14. Hopp T, Reeve C (1996) An algorithm for computing the minimum covering sphere in any dimension. NIST 5831 Tech Rep, NIST
15. Kumar P, Mitchell JSB, Yıldırım A (2003) Computing core-sets and approximate smallest enclosing hyperspheres in high dimensions. ACM J Exp Alg 8(1)
16. Megiddo N (1984) Linear programming in linear time when the dimension is fixed. J ACM 31(1):114-127. DOI 10.1145/2422.322418
17. Megiddo N (1989) On the ball spanned by balls. Discrete Comput Geom 4:605-610
18. Nesterov YE, Todd JE (1998) Primal-dual interior-point methods for self-scaled cones. SIAM J Optimization 8:324-364. DOI 10.1137/S1052623495290209
19. Ritter J (1990) An efficient bounding sphere. In: Glassner A (ed) Graphics Gems, pp 301-303. Academic Press
20. Sylvester JJ (1857) A question in the geometry of situation. Quarterly J Mathematics 1:79
21. Szabo T, Welzl E (2001) Unique sink orientations of cubes. Proc 42nd Ann Sympos Foundat Comp Sci (FOCS), pp 547-555
22. Welzl E (1991) Smallest enclosing disks (balls and ellipsoids). In: Maurer H (ed) New Results and New Trends in Computer Science, LNCS 555:359-370
23. Wu X (1992) A linear-time simple bounding volume algorithms. In: Kirk D (ed) Graphics Gems III, pp 301-306. Academic Press
24. Zhou G, Sun J, Toh KC (2003) Efficient algorithms for the smallest enclosing ball problem in high dimensional space. AMS Fields Institute Communications 37