Approximating Smallest Enclosing Disks

Frank Nielsen[*] Richard Nock[†]

Abstract

We describe a short and fast algorithm for finding arbitrarily fine approximations of the smallest enclosing disk of a planar point or disk set. Experimental results of an implementation are presented.

1 Introduction

The smallest enclosing disk (SED for short) problem dates back to 1857 when J. J. Sylvester [7] first asked for the smallest disk enclosing n points on the plane. Although $O(n \log n)$-time algorithms were designed for the planar case in the early 1970s [4, 6], its complexity was only settled in 1984 with N. Megiddo's first linear time algorithm [3] for solving linear programs in fixed dimension. Unfortunately, these algorithms exhibit a large constant hidden in the big-Oh notation and do not perform so well in practice. E. Welzl [8] developed a simple recursive $\tilde{O}(n)$ randomized algorithm for point sets, called "move-to-front" heuristic, that is often used by practitioners (see Section 5). Recently, Fischer et al. [2, 1] described a pivoting scheme resembling the simplex method for linear programming that, despite no theoretical time bounds (besides proven termination), can tackle exactly problems in large dimensions for ball sets. Computing smallest enclosing disks are useful for metrology, machine learning and computer graphics problems. Fast constant approximation heuristics are popular in computer graphics [5]. Our paper aims at designing a fast deterministic (*i.e.,* worst-case time bounded) approximation algorithm that is suitable for real-time demanding applications. Since they gain in speed as the precision decreases, approximation algorithms are well suited for such purposes. Our simple implementation for point/disk sets is a mere 30-line code which does not require to compute the basic primitive of the smallest disk enclosing three disks. In fact, surprisingly, we exhibit a robust approximation algorithm using only algebraic predicates of degree 2 using integer arithmetic. Moreover, as shown in Section 5, our floating-point implementation outperforms or fairly competes with traditional methods while guaranteeing worst-case termination time.

2 Approximating SED of Points

Let $\mathcal{P} = \{P_i = (x_i, y_i), i \in \{1, ..., n\}\}$ be a set of n planar points. We use notations $x(P_i) = x_i$ and $y(P_i) = y_i$ to refer to point coordinates. Let $Disk(C^*, r^*)$ be the unique (see [8]) smallest enclosing disk of $\mathcal{P}$ of center point C^* (also called circumcenter or Euclidean 1-center) and minimum radius r^*. We want to compute a $(1+\epsilon)$-approximation, that is, a disk $Disk(C, r)$ such that $r \leq (1+\epsilon)r^*$ and $\mathcal{P} \subseteq Disk(C, r)$.

2.1 Solving Decision Problems

Our approximation algorithm proceeds by solving dual piercing *decision problems* (DPs for short; see Figure 1): given a set of disks $\mathcal{B}(r) = \{B_i = Disk(P_i, r), i \in \{1, ..., n\}\}$, determine whether $\cap\mathcal{B}(r) = \cap_{i=1}^n B_i = \emptyset$ or not. We relax the 1-piercing point problem to that of finding a common piercing ϵr^*-disk (*i.e.*, a disk of radius ϵr^*): Namely, report whether there exists a disk $B = Disk(C, \epsilon r^*)$ such that $B \subseteq \cap\mathcal{B}(r)$ or not.

Lemma 1 *Observe that for $r \geq r^*$, there exists a disk B of radius $r(B) = r - r*$ centered at $C(B) = C^*$ fully contained inside $\cap\mathcal{B}$.*

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 $B_i \subseteq \mathrm{Disk}(C^*, r^*), \forall i \in \{1, 2, ..., n\}$, 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 $\forall i \in \{1, 2, ..., n\}, \forall 0 \leq r' \leq (r - r_i) - d_2(P_i, C^*), \mathrm{Disk}(C^*, r') \subseteq B_i(r)$. Thus, if we ensure that $r' \leq r - \max_{i=1}^n(r_i + d_2(P_i, C^*))$, then $\mathrm{Disk}(C^*, r') \subseteq \cap\mathcal{B}(r)$. From ineq. $(\star)$, we choose $r' = r - r^*$ and obtain the lemma (see Figure 1). ∎

[*]Sony Computer Science Laboratories.
E-mail: `Frank.Nielsen@acm.org`

[†]Université Antilles-Guyane, DSI GRIMAAG.
E-mail: `rnock@martinique.univ-ag.fr`

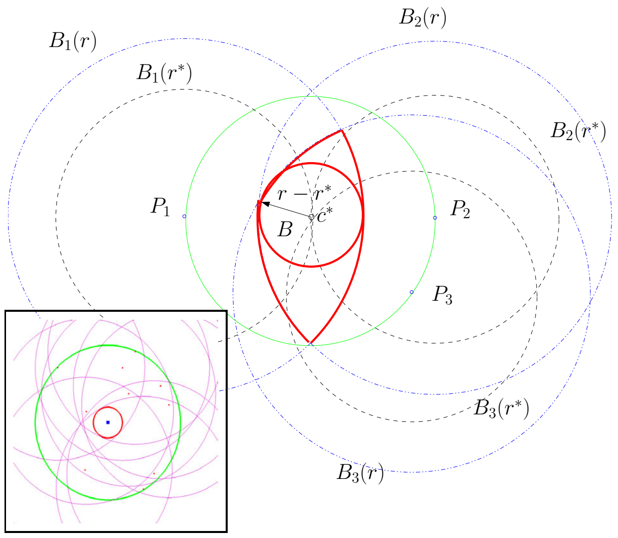

Figure 1: Covering/piercing duality principle. Points P_1, P_2, P_3 are associated to corresponding disks $B_1(r), B_2(r), B_3(r)$ such that $C(B_i(r)) = P_i$ and $r(B_i(r)) = r$ for $i \in \{1,2,3\}$. We have $B_1(r^*) \cap B_2(r^*) \cap B_3(r^*) = \{C^*\}$. For $r \geq r^*$, there exists a disk of radius $r - r^*$ fully contained in $B_1(r) \cap B_2(r) \cap B_3(r)$. Inset: SEB of 10 points (green), $DP(1.2r^*)$ (purple) and enclosed ball (red) of radius $0.2r^*$.

Let $[x_m, x_M]$ 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: $x_m = \max_i(x_i) - r$, $x_M = \min_i(x_i) + r$. 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 (and not ϵr^*-disk pierceable). Otherwise, the algorithm proceeds by dichotomy (see Figure 2). Let $e = \frac{x_m + x_M}{2}$ and let L denotes the vertical line $L : x = e$. Denote by $\mathcal{B}_L = \{B_i \cap L | i \in \{1, ..., n\}\}$ the set of n y-intervals obtained as the intersection of the disks of $\mathcal{B}$ with line L. We check whether $\mathcal{B}_L = \{B_i \cap L = [a_i, b_i] | i \in \{1, ..., n\}\}$ is 1-pierceable or not. Since $\mathcal{B}_L$ is a set of n y-intervals, we just need to check whether $\min_i b_i \geq \max_i a_i$ or not. If $\cap \mathcal{B}_L \neq \emptyset$, then we have found a point $(e, \min_i b_i)$ in the intersection of all disks 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 we 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 disks B_1 and B_2). Otherwise, $B_1 \cap B_2 \neq \emptyset$ and we branch on the side where $x_{B_1 B_2} = \frac{x(C(B_1)) + x(C(B_2))}{2}$ lies. At each stage

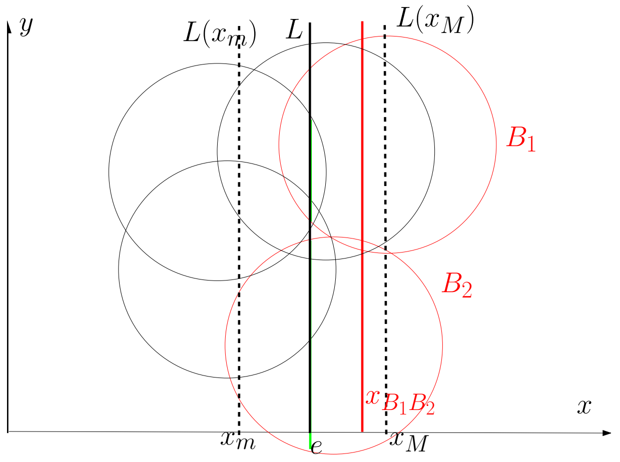

Figure 2: A recursion step: $L : x = e$ intersects all disks. Two y-intervals do not intersect on L. We recurse on x-range $[e, x_M]$.

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 disk* of radius ϵr is contained in $\cap \mathcal{B}$. (Indeed if such a disk 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 $2r^* \geq r \geq \frac{r^*}{2}$.

2.2 Dichotomy Search

Finding the minimum enclosing disk radius amounts to find the smallest value $r \in \mathbb{R}$ such that $\cap \mathcal{B}(r) \neq \emptyset$. That is $r^* = \mathrm{argmin}_r \cap \mathcal{B}(r) \neq \emptyset$. We seek for an $(1 + \epsilon)$-approximation of the minimum enclosing disk of points by doing a straightforward dichotomic process on relaxed decision problems. We always keep a solution interval $[a, b]$ where r^* lies, such that at any stage we have $\cap \mathcal{B}(a - \frac{\epsilon r^*}{2}) = \emptyset$ and $\cap \mathcal{B}(b) \neq \emptyset$. W.l.o.g., let P_1 denote the leftmost x-abscissæ point of $\mathcal{P}$ and let $P_2 \in \mathcal{P}$ be the maximum distance point of $\mathcal{P}$ from P_1. We have $r = d_2(P_1, P_2) \geq r^*$ (since $\mathcal{P} \subseteq \mathit{Disk}(P_1, r)$). But $d_2(P_1, P_2) \leq 2r^*$ since both P_1 and P_2 are contained inside the unique smallest enclosing disk of radius r^*. Thus we have $r^* \in [\frac{r}{2}, r]$.

We initialize the range by choosing $a = \frac{r}{2} \leq r^*$ and $b = r \leq 2r^*$. Then we solve the $\frac{\epsilon}{4}r$-disk piercing problem with disks of radius $e = \frac{a+b}{2}$. If we found a common piercing point for $\cap\mathcal{B}(e)$ then we recurse on $[a,e]$. Otherwise we recurse on $[e,b]$. We stop as soon as $b - a \leq \epsilon\frac{r}{4}$. (Therefore after $O(\log_2 \frac{1}{\epsilon})$ iterations since the initial range width $b - a \leq r^*$). At any stage, we assert that $\cap\mathcal{B}(a - \frac{\epsilon r}{4}) = \emptyset$ (by answering that $\cap\mathcal{B}(a)$ does not contain any disk of radius $\frac{\epsilon r}{4}$) and $\mathcal{B}(b) \neq \emptyset$. At the end of the recursion process, we get an interval $[a - \frac{\epsilon r}{4}, b]$ where r^* lies in. Since $b - a \leq \epsilon\frac{r}{4} \leq \epsilon\frac{r^*}{2}$ and $|r^* - a| < \frac{\epsilon r}{4} \leq \frac{\epsilon r^*}{2}$ (because $\mathcal{B}(a - \frac{\epsilon r}{4}) = \emptyset$), we get: $b \leq r^* + 2\epsilon\frac{r}{4}$. Since $r \leq 2r^*$, we obtain a $(1+\epsilon)$-approximation of the minimum enclosing disk of the point set. Thus, by solving $O(\log_2 \frac{1}{\epsilon})$ decision problems, we obtain a $O(n \log_2^2 \frac{1}{\epsilon})$-time deterministic $(1+\epsilon)$-approximation algorithm.

2.3 Bootstrapping

We bootstrap the previous 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 $Disk(C,r)$ be a $(1+\epsilon)$-approximation enclosing disk. 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 disk 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 $\sum_{i=0}^{\lceil \log_2 \frac{1}{\epsilon} \rceil} O(n \log_2 2) = O(n \log_2 \frac{1}{\epsilon})$. We get the following theorem:

Theorem 1 *A $(1+\epsilon)$-approximation of the minimum enclosing disk of a set of n points on the plane can be computed efficiently in $O(n \log_2 \frac{1}{\epsilon})$ deterministic time.*

3 Predicate Degree

Predicates are the basic computational atoms of algorithms that are related to their numerical stabilities. In the exact smallest enclosing disk algorithm [8], the so-called *InCircle* containment predicate of algebraic degree 4 is used on Integers. Since we only use $\sqrt{\cdot}$ function to determine the sign of algebraic numbers,

```
Data   : A set $\mathcal{S} = \{S_i = (x_i, y_i) | i \in \{1,...,n\}\}$ of $n$ points and an approximation factor $\epsilon$.
Result : $Disk((x,y),r)$: a $(1+\epsilon)$-approximation of the minimum enclosing disk of $\mathcal{S}$. That is we have $r^* \leq r \leq (1+\epsilon)r^*$, where $r^*$ is the minimum radius of enclosing disks.

$xmin = \min_{i \in \{1,...,n\}} x_i$;
$xmax = \max_{i \in \{1,...,n\}} x_i$;
$d_1 = \max_{i \in \{1,...,n\}} ||S_i - S_1||$;
$b = d_1$;
$a = \frac{d_1}{2}$;
$\epsilon \leftarrow \frac{1}{4}(b-a)\epsilon$;
$pierceable = false$;
$qdisjoint = false$;
while $b - a > \epsilon$ do
    $r = \frac{a+b}{2}$;
    $x_M = xmin + r$;
    $x_m = xmax - r$;
    $pierceable = false$;
    while $x_M - x_m \geq \epsilon$ and $\neg pierceable$ and $\neg qdisjoint$ do
        $l = \frac{x_M + x_m}{2}$;
        $y_m = \max_{i \in \{1,...,n\}} y_i - \sqrt{r^2 - (l - x_i)^2}$;
        $m = \mathrm{argmax}_{i \in \{1,...,n\}} y_i - \sqrt{r^2 - (l - x_i)^2}$;
        $y_M = \min_{i \in \{1,...,n\}} y_i + \sqrt{r^2 - (l - x_i)^2}$;
        $M = \mathrm{argmin}_{i \in \{1,...,n\}} y_i + \sqrt{r^2 - (l - x_i)^2}$;
        if $y_M \geq y_m$ then
            $x = l$;
            $y = \frac{y_m + y_M}{2}$;
            $pierceable = true$;
        else
            // $m$ and $M$ are arg indices of $y_m$ and $y_M$;
            if $||S_m - S_M|| > 2(r - \epsilon)$ then
                $qdisjoint = true$;
            else
                if $\frac{x_m + x_M}{2} > l$ then
                    $x_m = l$;
                else
                    $x_M = l$;
                end
            end
        end
    end
    if $pierceable$ then
        $b = r$;
    else
        if $qdisjoint$ then
            $a = \frac{||S_m - S_M||}{2} + \epsilon$ ;
        else
            $a = r$;
        end
    end
end
```

Algorithm 1: $(1+\epsilon)$-approximation algorithm of the minimum enclosing disk of a 2d point set.

Method/Distribution	□ Square max	⊙ Ring max	□ Square avg	⊙ Ring avg
Eberly ($\epsilon = 10^{-5}$)	0.7056	0.6374	0.1955	0.2767
Ritter ($\epsilon > 0.15$)	0.0070	0.0069	0.0049	0.0049
ASED ($\epsilon = 10^{-2}$)	0.0343	0.0338	0.0205	0.0286
ASED ($\epsilon = 10^{-3}$)	0.0515	0.0444	0.0284	0.0405
ASED ($\epsilon = 10^{-4}$)	0.0646	0.0617	0.0392	0.0449
ASED ($\epsilon = 10^{-5}$)	0.0719	0.0726	0.0473	0.0527

Table 1: Timings. Experiments done on 1000 trials for point sets of size 100000. Maximum (max) and average (avg) running times are in fractions of a second.

all computations can be done on Rationals using algebraic degree 2 (also observed in [1]). We show how to replace the predicates of algebraic degree[1] 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^*-disk 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 $O(n \frac{1}{\epsilon} \log_2 \frac{1}{\epsilon})$. 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 the following theorem:

Theorem 2 *A $(1 + \epsilon)$-approximation of the minimum enclosing disk of a set of n points on the plane can be computed in $O(n \log_2 \frac{1}{\epsilon})$ time using Integer arithmetic with algebraic predicates InCircle of degree 2.*

4 Approximating SED of Disks

Our algorithm extends straightforwardly for sets of disks. Consider a set of n planar disks $\mathcal{D} = \{D_1, ..., D_n\}$ with $C(D_i) = P_i = (x_i, y_i)$ and $r(D_i) = r_i$. Let $\mathcal{B}(r) = \{B_i | C(B_i) = P_i \text{ and } r(B_i) = r - r_i\}$. Using the dual piercing principle, we obtain that $r^* = \text{argmin}_{r \in \mathbb{R}} \cap \mathcal{B}(r) \neq \emptyset$. (We have $C^* = \cap\mathcal{B}(r^*)$.) Observe also that $r^* \geq \max_{i \in \{1,...,n\}} r_i$. Initialization is done by choosing $b = r_1 + \max_{i \in \{1,...,n\}}(d_2(P_1, P_i) + r_i)$ and $a = \frac{b}{2}$. We now let:

$$x_{B_1 B_2} = x_{B_1} + \frac{r_2^2 - r_1^2 + (r_1 + r_2)^2}{2(r_1 + r_2)^2}(x_{B_2} - x_{B_1}).$$

5 Experimental Results

We compare our implementation (see pseudo-code on the left) with D. H. Eberly's C++ implementation using double types that guarantees precision $\epsilon = 10^{-5}$ and has expected running time $10n$ but no known worst-case bound better than $O(n!)$. We also compare our code with J. Ritter's constant approximation ($\epsilon \simeq 15\%$) single-pass greedy heuristic used in game programming [5]. Timings are obtained on an Intel Pentium(R) 4 1.6 GHz with 1 GB of memory for points uniformly distributed inside a unit square (□) and inside a unit ring of width 0.01 (⊙). Table 1 reports our timings. The experiments show that over a thousand square/ring random point sets, our algorithm maximum time is much smaller than that of D. H. Eberly's (in addition, this latter algorithm requires $\tilde{O}(\log_2^3 n)$ calls [8] to the expensive basic primitives of computing the circle passing through three points).

We are grateful to Jason Hughes for pointing out a

[1]Comparing expressions $y_1 + \sqrt{r^2 - (l - x_1)^2} > y_2 + \sqrt{r^2 - (l - x_2)^2}$ is of degree 4 for Integers. Indeed, by isolating and removing the square roots by successive squaring, the predicate sign is the same as $(2r^2 - (l - x_1)^2 - (l - x_2)^2)^2 > 4(r^2 - (l - x_1)^2)(r^2 - (l - x_2)^2)$. The last polynomial has highest monomials of degree 4.

typographic error that occurred when converting our C code to LaTeX.

References

[1] K. Fischer and B. Gärtner. The smallest enclosing ball of balls: combinatorial structure and algorithms. In *Proceedings of the 19th Conference on Computational Geometry*, pages 292–301. ACM Press, 2003.

[2] K. Fischer, B. Gärtner, and M. Kutz. Fast smallest-enclosing-ball computation in high dimensions. In *Proceedings of the 11th Annual European Symposium on Algorithms*, *LNCS* 2832, pages 630–641. Springer-Verlag, 2003.

[3] N. Megiddo. Linear programming in linear time when the dimension is fixed. *Journal of the ACM*, Vol. 31(1), pages 114–127, January 1984.

[4] F. P. Preparata and M. I. Shamos. *Computational Geometry: An Introduction.* Springer-Verlag, New York, NY, 1985.

[5] J. Ritter. An efficient bounding sphere. In A. Glassner, editor, *Game Programming Gems*, pages 301–303. Academic Press, Boston, 1990.

[6] S. Skyum. A simple algorithm for computing the smallest enclosing circle. *Information Processing Letters*, Vol. 37, pages 121–125, 1991.

[7] J. J. Sylvester. A question in the geometry of situation. *Quarterly Journal of Mathematics*, Vol. 1, page 79, 1857.

[8] E. Welzl. Smallest enclosing disks (balls and ellipsoids). In H. Maurer (Ed.), *New Results and New Trends in Computer Science*, *LNCS* 555, pages 359–370. Springer-Verlag, 1991.