Fast Algorithms for Computing the Smallest k-Enclosing Circle[*]

Sariel Har-Peled[†] Soham Mazumdar[‡]

April 4, 2006

Abstract

For a set P of n points in the plane and an integer $k \leq n$, consider the problem of finding the smallest circle enclosing at least k points of P. We present a randomized algorithm that computes in $O(nk)$ expected time such a circle, improving over previously known algorithms. Further we present a linear time δ-approximation algorithm that outputs a circle that contains at least k points of P and has radius less than $(1+\delta)r_{opt}(P,k)$, where $r_{opt}(P,k)$ is the radius of the minimum circle containing at least k points of P. The expected running time of this approximation algorithm is $O\big(n + n \cdot \min\big(\frac{1}{k\delta^3}\log^2\frac{1}{\delta}, k\big)\big)$.

1. Introduction

Shape fitting, a fundamental problem in computational geometry [AHV04], computer vision, machine learning, data mining and many other areas, is concerned with finding the best shape which "fits" a given input. This problem has attracted a lot of research both for the exact and approximation versions; see [BE97, AHV04] and references therein.

Furthermore, solving such problems in the real world is quite challenging, as noise in the input is omnipresent and one has to assume that some of the input points are noise, and as such should be ignored. See [Cha05, DLSS95, EE94, Mat95b] for some recent relevant results. Unfortunately, under such noisy conditions, the shape fitting problem becomes notably harder.

[*]A preliminary version of the paper appeared in *Proc. 11th Annu. European Sympos. Algorithms*, volume 2832 of *Lect. Notes in Comp. Sci.*, pages 278–288. Springer-Verlag, 2003.

[†]Department of Computer Science; University of Illinois; 201 N. Goodwin Avenue; Urbana, IL, 61801, USA; sariel@illinois.edu; http://sarielhp.org/. Work on this paper was partially supported by a NSF CAREER award CCR-0132901.

[‡]Department of Computer Science; University of Illinois; 201 N. Goodwin Avenue; Urbana, IL, 61801, USA; smazumda@cs.uiuc.edu.

An important class, of shape fitting problems, involves finding an optimal k point subset of a set of n points based on some optimizing criteria. The optimizing criteria could be the smallest convex hull volume, the smallest enclosing ball, the smallest enclosing box, the smallest diameter among others [EE94, AST94].

An interesting problem in this class is the computation of the smallest circle containing k points of a given set of n points in the plane. The initial approaches to solving this problem involved first constructing the order-k Voronoi diagram, followed by a search in all or some of the Voronoi cells. The best known algorithm to compute the order-k Voronoi diagram has time complexity $O(nk \log n + n \log^3 n)$ [ABMS98]. Eppstein and Erickson [EE94] observed that instead of Voronoi cells, one can work with $O(k)$ nearest neighbors to each point. The resulting algorithm has a running time of $O(nk \log n + nk \log^2 k)$ and space complexity $O(nk + k^2 \log k)$. Using the technique of parametric search, Efrat *et al.* [ESZ94] solved the problem in time $O(nk \log^2 n)$ and space $O(nk)$. Finally, using a suitable randomized search, Matoušek gave a simple algorithm which uses $O(nk)$ space and has $O(n \log n + nk)$ expected running time [Mat95a].

We revisit this classical problem, and present an algorithm with $O(nk)$ expected running time, that uses $O(n + k^2)$ space. The main reason why this result is interesting is because it beats the lower bound of $\Omega(n \log n)$ on the running time for small k, which follows from element uniqueness in the comparison model. We achieve this by using randomization and the floor function (interestingly enough, this is also the computation model used by Matoušek [Mat95a]). Despite this somewhat small gain, removing the extra $\log n$ factor from the running time was a non-trivial undertaking, requiring some new ideas.

Our main technical contribution is a *linear* time 2-approximation algorithm, described in Section 4. This improves upon the previous best result of Matoušek [Mat95a] that runs in $O(n \log n)$ time. Using our algorithm and the latter half of the algorithm of Matoušek (with some minor modifications), we get the new improved exact algorithm. Finally, in Section 5, we observe that from the 2-approximation algorithm one can get a $(1 + \delta)$-approximation algorithm with running time linear in n and polynomial in $1/\delta$.

2. Preliminaries

For r a real positive number and a point $p = (x, y)$ in $\mathbb{R}^2$, define $\mathrm{G}_r(p)$ to be the point $(\lfloor x/r \rfloor r, \lfloor y/r \rfloor r)$. We call r the *width* of the *grid* G_r. Observe that G_r partitions the plane into square regions, which we call grid *cells*. Formally, for any $i, j \in \mathbb{Z}$, the intersection of the half-planes $x \geq ri$, $x < r(i+1)$, $y \geq rj$ and $y < r(j+1)$ is said to be a grid cell. Further we define a *grid cluster* as a block of 3×3 contiguous grid cells. For a circle D, we denote by $\mathrm{radius}(D)$ the *radius* of D.

For a point set P, and parameter r, the partition of P into subsets by the grid G_r, is denoted by $\mathrm{G}_r(P)$. More formally, two points $p, q \in P$ belong to the same set in the partition $\mathrm{G}_r(P)$, if both points are being mapped to the same grid point or equivalently belong to the same grid cell.

Let $\mathtt{gd}_r(P)$ denote the maximum number of points of P mapped to a single point by the mapping G_r. Define $\texttt{depth}(P, r)$ to be the maximum number of points of P that a circle of radius r can contain. Let $D_{opt}(P, k)$ be a circle of minimum radius which contains k points of P, and let $r_{opt}(P, k)$ denote the radius of $D_{opt}(P, k)$.

The above notation is a slight variant on definitions of Matoušek [Mat95a]. Using simple packing arguments one can prove the following results (using similar argumentation to [Mat95a]):

Lemma 2.1. *For any point set P, and $r > 0$, we have: (i) For any real number $A > 0$, it holds $\texttt{depth}(P, Ar) \le (A+1)^2 \texttt{depth}(P, r)$, (ii) $\texttt{gd}_r(P) \le \texttt{depth}(P, r) \le 9\texttt{gd}_r(P)$, (iii) if $r_{opt}(P, k) \le r \le 2r_{opt}(P, k)$ then $\texttt{gd}_r(P) \le 5k$, and (iv) Any circle of radius r is covered by at least one grid cluster in G_r.*

Definition 2.2 (Gradation). Given a set P of n points, a *sampling sequence* $(S_m, \ldots, S_1)$ of P is a sequence of subsets of P, such that (i) $S_1 = P$, (ii) S_i is formed by picking each point of S_{i-1} with probability $1/2$, and (iii) $|S_m| \le n/\log n$, and $|S_{m-1}| > n/\log n$. The sequence $(S_m, S_{m-1}, \ldots, S_1)$ is called a *gradation* of P.

Lemma 2.3. *Given P, a* sampling sequence *can be computed in expected linear time.*

Proof: Observe that the sampling time is $O(\sum_{i=1}^{m} |S_i|)$, where m is the length of the sequence. Also observe that $\mathbb{E}[|S_1|] = |S_1| = n$ and

$$\mathbb{E}\big[|S_i|\big] = \mathbb{E}\Big[\mathbb{E}\big[|S_i| \mid |S_{i-1}|\big]\Big] = \mathbb{E}\left[\frac{|S_{i-1}|}{2}\right] = \frac{1}{2}\,\mathbb{E}\big[|S_{i-1}|\big].$$

Now by induction, we get

$$\mathbb{E}\big[|S_i|\big] = \frac{n}{2^{i-1}}.$$

Thus, the running time is $O(\mathbb{E}[\sum_{i=1}^{m} |S_i|]) = O(n)$. ■

3. A Slow 2-Approximation Algorithm

In this section, we develop a $O(n \cdot (n/k)^2)$ time 2-approximation algorithm that would be used as a subroutine in our faster algorithm, presented in Section 4. Surprisingly, the algorithm in Section 4 merely requires a 2-approximation algorithm with a running time of $O(n \cdot (n/k)^c)$, where c is a constant.

3.1. A Deterministic Algorithm

Let P be a set of n points in the plane. Compute a set of $m = O(n/k)$ horizontal lines $h_1, \ldots, h_m$ such that between two consecutive horizontal lines, there are at most $k/4$ points of P in the strip they define. This can be easily done in $O(n \log(n/k))$ time using deterministic median selection together with recursion. Similarly, compute a set of vertical lines $v_1, \ldots, v_m$, such that between two consecutive lines, there are at most $k/4$ points of P.

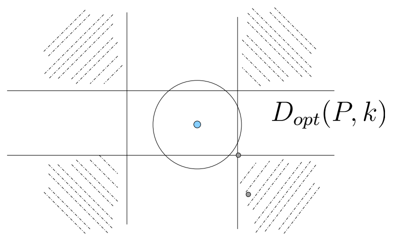

Figure 1: $D_{opt}(P, k) \cap X \neq \phi$.

Consider the (non-uniform) grid G induced by $h_1, \ldots, h_m$ and $v_1, \ldots, v_m$. Let X be the set of all intersection points of G. We claim that $D_{opt}(P, k)$ contains at least one point of X. Indeed, consider the center u of $D_{opt}(P, k)$, and let c be the cell of G that contains u. Clearly, if $D_{opt}(P, k)$ does not cover any of the four vertices of c, then it can cover only points in the vertical strip of G that contains c, and only points in the horizontal strip of G that contains c. See Figure 1. However, each such strip contains at most $k/4$ points. It follows that $D_{opt}(P, k)$ contains at most $k/2$ points of P, a contradiction. Thus, $D_{opt}(P, k)$ must contain a point of X. For every point $p \in X$, compute the smallest circle centered at p that contains k points of P. Clearly, for a point $q \in X \cap D_{opt}(P, k)$, this yields the required 2-approximation. We summarize as follows:

Lemma 3.1. *Given a set P of n points in the plane, and parameter k, one can compute in $O(n(n/k)^2)$ deterministic time, a circle D that contains k points of P, and* $\mathrm{radius}(D) \leq 2r_{opt}(P, k)$.

Corollary 3.2. *Given a set of P of n points and a parameter $k = \Omega(n)$, one can compute in linear time, a circle D that contains k points of P and* $\mathrm{radius}(D) \leq 2r_{opt}(P, k)$.

3.2. A Randomized Algorithm

Let R be a random sample from P, generated by choosing every point of P with probability $1/k$. Next, compute for every $p \in R$, the smallest circle centered at p containing k points of P. Overall, this takes $O(n(n/k))$ expected time, as $\mathbb{E}[|R|] = n/k$ and computing the smallest circle for every point takes linear time using median selection. Let r be the minimum radius computed.

Let X be the set of k points of P covered by $D_{opt}(P, k)$. We have

$$\mathbb{P}[X \cap R \neq \emptyset] = 1 - \left(1 - \frac{1}{k}\right)^k \geq 1 - \frac{1}{e}.$$

If R contains a point covered by $D_{opt}(P, k)$, then r is a 2-approximation to $r_{opt}(P, k)$. We need to verify that this indeed occurred.

To this end, we maintain a hash table of non-empty cells in the grid G_r. In the beginning this hash table is empty. Next, we scan the points one by one. For every point, we check whether $\mathrm{G}_r(p)$ is already registered in the hash table. If it is not in the hash table, then we add $\mathrm{G}_r(p)$ to our list of non-empty grid cells, create a new counter c for this cell (initialized to zero), and store $(\mathrm{G}_r(p), c)$ in the hash table. If $\mathrm{G}_r(p)$ was already in the hash table, then we just increment the counter associated with $\mathrm{G}_r(p)$.

In particular, in the end of this scan, we have an explicit list of all the non-empty cells in the grid. Furthermore, we can now scan the list, and generate a list of all nonempty clusters in the grid. This is because every non-empty cell in the grid, induces 9 nonempty clusters. Furthermore, it is easy to remove repetitions from the generated list of clusters by hashing. Thus, if we are interested in all the clusters of weight at least k, then we can scan this list, compute the weight of each cluster, and keep only the ones that are heavy enough.

Thus, to verify that the algorithm succeeded, compute the grid G_r, and snap the points of P to G_r, as described above. If any cluster of G_r contains more than, say, $45k$ points, then r is clearly too large, and the algorithm failed, by Lemma 2.1 (iii). In such a case, we just run the above procedure again. Otherwise, we run on each cluster of G_r that contains at least k points, the algorithm of Corollary 3.2. Let D be the smallest circle computed that contains k points of P. By Lemma 2.1 there is a cluster covering the optimal circle, and as such D is a 2-approximation to $D_{opt}(P, k)$. The overall running time of this stage is $O((n/k)k) = O(n)$. This is because there are at most $9n/k$ clusters containing more than k points, as every point participates in at most 9 clusters. Further, each such cluster contains less than $45k$ points.

The algorithm succeeds with constant probability and is repeated till it succeeds, as such it follows that the expected running time of our algorithm is $O(n(n/k))$.

Lemma 3.3. *Given a set P of n points in the plane, and parameter k, one can compute in expected $O(n(n/k))$ time, a circle D that contains k points of P, and* $\mathrm{radius}(D) \leq 2r_{opt}(P, k)$.

We refer to the algorithm of Lemma 3.3 as **ApproxHeavy**(P, k).

Remark 3.4. For a point set P of n points, the radius r returned by the algorithm of Lemma 3.3 is the distance between a pair of points of P. As such, a grid G_r computed using this distance is one of $O(n^3)$ possible grids. Indeed, a circle is defined by the distance between a vertex of the non-uniform grid of Lemma 3.1, and a point of P. A vertex of such a grid is determined by two points of P, through which the vertical and horizontal line passes. Thus, there are $O(n^3)$ such triples.

In an earlier version of this paper [HS03], we had a considerably more involved algorithm that performed the 2-approximation in $O(n \log(n/k))$ time. However, for our purposes, any one of the slow (by simpler) algorithms of Lemma 3.1 or Lemma 3.3, are sufficient. See [HS03] for details.

```
Grow($P_i$,$r_{i-1}$,$k$)
        Output: $r_i$
    begin
        $\mathrm{G}_{i-1} \leftarrow \mathrm{G}_{r_{i-1}}(P_i)$
        for every grid cluster $c \in \mathrm{G}_{i-1}$ with $|c \cap P_i| \geq k$ do
            $P_c \leftarrow c \cap P_i$
            $r_c \leftarrow$ ApproxHeavy($P_c, k$)
                // ApproxHeavy is the algorithm of Lemma 3.3
            We have $r_{opt}(P_c, k) \leq r_c \leq 2r_{opt}(P_c, k)$,

        return  minimum $r_c$ computed.
    end
```

Figure 2: Algorithm for the ith round.

4. A 2-Approximation in Linear Time

4.1. Description

As in the previous sections, we construct a grid which partitions the points into small ($O(k)$ sized) groups. The key idea behind speeding up the grid computation is to construct the appropriate grid over several rounds. Specifically, we start with a small set of points as seed and construct a suitable grid for this subset. Next, we incrementally insert the remaining points, while adjusting the grid width appropriately at each step.

Let $\mathcal{P} = (P_1, \ldots, P_m)$ be a gradation of P (see Definition 2.2), where $P = P_m$ and $|P_1| \geq \max(k, n/\log n)$ (i.e. if $k \geq n/\log n$ start from the first set in $\mathcal{P}$ that has more than k elements). The sequence $\mathcal{P}$ can be computed in expected linear time as shown in Lemma 2.3.

If $k < n/\log n$, then $|P_1| = O(n/\log n)$, and we use the 2-approximation algorithm of [Mat95a], and obtain a length r_1 such that $r_{opt}(P_1, k) \leq r_1 \leq 2r_{opt}(P_1, k)$ and $\mathtt{gd}_{r_1}(P_1) \leq 5k$. The set P_1 is the seed mentioned earlier. Observe that it takes $O(|P_1| \log |P_1|) = O(n)$ time to perform this step.

If $|P_1| \geq k \geq n/\log n$, then $|P_1| = O(k)$, and we can compute r_1, in $O(|P_1| (|P_1|/k)^2) = O(k) = O(n)$ time, using Lemma 3.1.

The remaining algorithm works in m rounds, where m is the length of the sequence $\mathcal{P}$. At the end of the ith round, we have a distance r_i such that $\mathtt{gd}_{r_i}(P_i) \leq 5k$, and there exists a grid cluster in G_{r_i} containing more than k points of P_i and $r_{opt}(P_i) \leq r_i$.

At the ith round, we first construct a grid G_{i-1} for points in P_i using r_{i-1} as grid width. We know that there is no grid cell containing more than $5k$ points of P_{i-1}. As such, intuitively, we expect every cell of G_{i-1} to contain at most $10k$ points of P_i, since $P_{i-1} \subseteq P_i$ was formed by choosing each point of P_i into P_{i-1} with probability $1/2$. (This is of course too good to be true, but something slightly weaker does hold.) Thus allowing us to use the slow algorithm of Lemma 3.3 on those grid clusters. Note that, for $k = \Omega(n)$, the algorithm

```
LinearApprox($P, k$)
    Output: $r$ - a 2-approximation to $r_{opt}(P, k)$
  begin
    Compute a gradation $\{P_1, \ldots, P_m\}$ of P as in Lemma 2.3
    $r_1 \leftarrow$ ApproxHeavy($P_1, k$)
        // ApproxHeavy is the algorithm of Lemma 3.3
          // which outputs a 2-approximation

    for $i \leftarrow 2$ to $m$ do
      $r_j \leftarrow$ Grow($P_i, r_{i-1}, k$)

    for every grid cluster $c \in \mathrm{G}_{r_m}$ with $|c \cap P| \geq k$ do
      $r_c \leftarrow$ ApproxHeavy($c \cap P, k$)

    return minimum $r_c$ computed over all clusters
  end
```

Figure 3: 2-Approximation Algorithm.

of Lemma 3.3 runs in expected linear time, and thus the overall running time is linear.

The algorithm used in the ith round is more concisely stated in Figure 2. At the end of the m rounds we have r_m, which is a 2-approximation to the radius of the optimal k enclosing circle of $P_m = P$. The overall algorithm is summarized in Figure 3.

4.2. Analysis

Lemma 4.1. *For $i = 1, \ldots, m$, we have $r_{opt}(P_i, k) \leq r_i \leq 2r_{opt}(P_i, k)$, and the heaviest cell in $\mathrm{G}_{r_i}(P_i)$ contains at most $5k$ points of P_i.*

Proof: Consider the optimal circle D_i that realizes $r_{opt}(P_i, k)$. Observe that there is a cluster c of $\mathrm{G}_{r_{i-1}}$ that contains D_i, as $r_{i-1} \geq r_i$. Thus, when **Grow** handles the cluster c, we have $D_i \cap P_i \subseteq c$. The first part of the lemma then follows from the correctness of the algorithm of Lemma 3.3.

As for the second part, observe that any grid cell of width r_i can be covered with 5 circles of radius $r_i/2$, and $r_i/2 \leq r_{opt}(P_i, k)$. It follows that each grid cell of $\mathrm{G}_{r_i}(P_i)$ contains at most $5k$ points. ■

Now we proceed to upper-bound the number of cells of $\mathrm{G}_{r_{i-1}}$ that contains "too many" points of P_i. Since each point of P_{i-1} was chosen from P_i with probability $1/2$, we can express this bound as a sum of independent random variables, and bound this using tail-bounds.

Definition 4.2. For a point set P, and parameters k and r, the *excess* of $\mathrm{G}_r(P)$ is

$$\mathcal{E}(P, k, \mathrm{G}_r) = \sum_{c \in \mathrm{Cells}(\mathrm{G}_r)} \left\lfloor \frac{|c \cap P|}{50k} \right\rfloor,$$

where $\mathrm{Cells}(\mathrm{G}_r)$ is the set of cells of the grid G_r.

Remark 4.3. The quantity $100k \cdot \mathcal{E}(P, k, \mathrm{G}_r)$ is an upper bound on the number of points of P in an heavy cell of $\mathrm{G}_r(P)$, where a cell of $\mathrm{G}_r(P)$ is *heavy* if it contains more than $50k$ points.

Lemma 4.4. *For any positive real t, the probability that $\mathrm{G}_{r_{i-1}}(P_i)$ has excess $\mathcal{E}(P_i, k, \mathrm{G}_{r_{i-1}}) \geq \alpha = t + 5\lceil \log(n) \rceil$, is at most 2^{-t}.*

Proof: Let $\mathfrak{G}$ be the set of $O(n^3)$ possible grids that might be considered by the algorithm (see Remark 3.4), and fix a grid $\mathsf{G} \in \mathfrak{G}$ with excess $M = \mathcal{E}(P_i, k, \mathsf{G}) \geq \alpha$.

Let $U = \{\{P_i \cap c\} \mid c \in \mathsf{G}, |P_i \cap c| > 50k\}$ be all the heavy cells in $\mathsf{G}(P_i)$. Furthermore, let $V = \bigcup_{X \in U} \psi(X, 50k)$, where $\psi(X, \nu)$ denotes an arbitrary partition of the set X into disjoint subsets such that each one of them contains ν points, except maybe the last subset that might contain between ν and $2\nu - 1$ points.

It is clear that $|V| = \mathcal{E}(P_i, k, \mathsf{G})$. From the Chernoff inequality, for any $S \in V$, we have $\mu = \mathbb{E}[|S \cap P_{i-1}|] \geq 25k$, and setting $\delta = 4/5$ we have

$$\mathbb{P}\big[|S \cap P_{i-1}| \leq 5k\big] \leq \mathbb{P}\big[|S \cap P_{i-1}| \leq (1-\delta)\mu\big] < \exp\left(-\mu\frac{\delta^2}{2}\right) = \exp\left(-\frac{25k(4/5)^2}{2}\right) < \frac{1}{2}.$$

Furthermore, $\mathsf{G} = \mathrm{G}_{r_{i-1}}$ implies that each cell of $\mathsf{G}(P_{i-1})$ contains at most $5k$ points. Thus we have

$$\mathbb{P}\big[\mathrm{G}_{r_{i-1}} = \mathsf{G}\big] \leq \prod_{S \in V} \mathbb{P}\big[|S \cap P_{i-1}| \leq 5k\big] \leq \frac{1}{2^{|V|}} = \frac{1}{2^M} \leq \frac{1}{2^\alpha}.$$

Since there are n^3 different grids in $\mathfrak{G}$, we have

$$\mathbb{P}\big[\mathcal{E}(P_i, k, \mathrm{G}_{r_{i-1}}) \geq \alpha\big] = \mathbb{P}\Big[\bigcup_{\substack{\mathsf{G} \in \mathfrak{G},\\ \mathcal{E}(P_i,k,\mathsf{G}) \geq \alpha}} (\mathsf{G} = \mathrm{G}_{r_{i-1}})\Big] \leq \sum_{\substack{\mathsf{G} \in \mathfrak{G},\\ \mathcal{E}(P_i,k,\mathsf{G}) \geq \alpha}} \mathbb{P}\big[\mathsf{G} = \mathrm{G}_{r_{i-1}}\big] \leq n^3 \frac{1}{2^\alpha} \leq \frac{1}{2^t}.$$

■

We next bound the expected running time of the algorithm **LinearApprox** by bounding the expected time spent in the ith iteration. In particular, let Y be the random variable which is the excess of $\mathrm{G}_{r_{i-1}}(P_i)$. In this case, there are at most Y cells which are heavy in $\mathrm{G}_{r_{i-1}}(P_i)$, and each such cell contains at most Yk points. Thus, invoking the algorithm of **ApproxHeavy** on such a heavy cell takes $O(Yk \cdot ((Yk)/k)) = O(Y^2 k)$ time. Overall, the running time of **Grow**, in the ith iteration, is $T(Y) = O(|P_i| + Y \cdot Y^2 k) = O(|P_i| + Y^3 k)$.

For technical reasons, we need to consider the light and heavy cases separately to bound Y.

4.2.1. The Light Case: $k < 4\log n$ .XX

We have that the expected running time is proportional to

$$\begin{aligned}\sum_{t=0}^{\lceil n/k\rceil} \mathbb{P}[Y=t]T(t) &= |P_i| + \mathbb{P}[0 \le Y \le 2\lceil \log n\rceil]T(2\lceil \log n\rceil) + \sum_{t=2\lceil \log n\rceil+1}^{n/k} \mathbb{P}[Y=t]T(t)\\ &\le |P_i| + T(2\lceil \log n\rceil) + \sum_{t=1}^{n/k} \frac{1}{2^t} T(t+2\lceil \log n\rceil)\\ &= O(|P_i|) + O(k\log^3 n) + \sum_{t=1}^{n/k} \frac{(t+2\lceil \log n\rceil)^3 k}{2^t}\\ &= O\big(|P_i| + k\log^3 n\big) = O(|P_i|),\end{aligned}$$

by Lemma 4.4 and since $T(\cdot)$ is a monotone increasing function.

4.2.2. The Heavy Case: $k \ge 4\log n$.

Lemma 4.5. *The probability that $\mathrm{G}_{r_{i-1}}(P_i)$ has excess larger than t, is at most 2^{-t}, for $k \ge 4\log n$.*

Proof: We use the same technique as in Lemma 4.4. By the Chernoff inequality, the probability that any $50k$ size subset of P_i would contain at most $5k$ points of P_{i-1}, is less than

$$\le \exp\left(-25k \cdot \frac{16}{25} \cdot \frac{1}{2}\right) \le \exp(-5k) \le \frac{1}{n^4}.$$

In particular, arguing as in Lemma 4.4, it follows that the probability that $\mathcal{E}(P_i, k, r_{i-1})$ exceeds t, is smaller than $n^3/n^{4t} \le 2^{-t}$. ■

Thus, if $k \ge 4\log n$, the expected running time of **Grow**, in the ith iteration, is at most

$$O\Big(\sum_{c\in \mathrm{G}_{r_{i-1}}} |c\cap P_i| \cdot \frac{|c \cap P_i|}{k}\Big) = O\left(|P_i| + \sum_{t=1}^{\infty} t\cdot \frac{(tk)^2}{k}\frac{1}{2^t}\right) = O\big(|P_i| + k\big) = O\big(|P_i|\big),$$

by Lemma 4.5.

4.2.3. Overall Running Time Analysis

Thus, by the above analysis and by Lemma 2.3, the total expected running time of **LinearApprox** inside the inner loop is $O(\sum_i |P_i|) = O(n)$. As for the last step, of computing a 2-approximation, consider the grid $\mathrm{G}_{r_m}(P)$. Each grid cell contains at most $5k$ points, and hence each grid cluster contains at most $45k$ points. Also the smallest k enclosing circle is contained in some grid cluster. In each cluster that contain more than k points, we use the algorithm of Corollary 3.2 and finally output the minimum over all the clusters. The overall running time is $O((n/k)k) = O(n)$ for this step, since each point belongs to at most 9 clusters.

Theorem 4.6. *Given a set P of n points in the plane, and a parameter k, one can compute, in expected linear time, a radius r, such that $r_{opt}(P, k) \leq r \leq 2r_{opt}(P, k)$.*

Once we compute r such that $r_{opt}(P, k) \leq r \leq 2r_{opt}(P, k)$, using the algorithm of Theorem 4.6, we apply the exact algorithm of Matoušek [Mat95a] to each cluster of the grid $\mathrm{G}_r(P)$ which contains more than k points.

Matoušek's algorithm has running time of $O(n \log n + nk)$ and space complexity $O(nk)$. Since r is a 2 approximation to $r_{opt}(P, k)$, each cluster has $O(k)$ points. Thus the running time of the exact algorithm in each cluster is $O(k^2)$ and requires $O(k^2)$ space. The number of clusters which contain more than k points is $O(n/k)$. Hence the overall running time is $O(nk)$, and the space used is $O(n + k^2)$.

Theorem 4.7. *Given a set P of n points in the plane and a parameter k, one can compute, in expected $O(nk)$ time, using $O(n + k^2)$ space, the radius $r_{opt}(P, k)$, and a circle $D_{opt}(P, k)$ that covers k points of P.*

5. From constant approximation to $(1+\delta)$-approximation

Let r be a 2-approximation to $r_{opt}(P, k)$. If we construct $\mathrm{G}_r(P)$, each grid cell contains less than $5k$ points of P (each grid cell can be covered fully by 5 circles of radius $r_{opt}(P, k)$). Furthermore, the smallest k-enclosing circle is covered by some grid cluster. We compute a $(1+\delta)$-approximation to the radius of the minimal k enclosing circle in each grid cluster and output the smallest among them. The technique to compute $(1+\delta)$-approximation when all the points belong to a particular grid cluster is as follows.

Let P_c be the set of points in a particular grid cluster with $k < |P_c| = O(k)$. Let R be a bounding square of the points of P_c. We partition R into a uniform grid G of width $10r\delta$. Next, we snap every point of P_c into the closest grid point of G, and let P'_c denote the resulting point set. Clearly, $|P'_c| = O(1/\delta^2)$. We guess the radius $r_{opt}(P_c, k)$ up to a factor of $1+\delta$ (there are only $O(\log_{1+\delta} 2) = O(1/\delta)$ possible guesses). Let r' be the current guess. We need to compute for each point p of P'_c, how many points of P'_c are contained in $D(p, r')$. This can be done in $O((1/\delta)\log(1/\delta))$ time per point, by constructing a quadtree over the points of P'_c. Thus, computing a $\delta/4$-approximation to the $r_{opt}(P'_c, k)$ takes $O((1/\delta^3)\log^2(1/\delta))$ time.

We repeat the above algorithm for all the clusters that have more than k points inside them. Clearly, the smallest circle computed is the required approximation. The running time is $O(n + n/(k\delta^3)\log^2(1/\delta))$. Putting this together with the algorithm of Theorem 4.6, we have:

Theorem 5.1. *Given a set P of n points in the plane, and parameters k and $\delta > 0$, one can compute, in expected*

$$O\left(n + n \cdot \min\left(\frac{1}{k\delta^3}\log^2\frac{1}{\delta}, k\right)\right)$$

time, a radius r, such that $r_{opt}(P, k) \leq r \leq (1+\delta)r_{opt}(P, k)$.

6. Conclusions

We presented a linear time 2-approximation algorithm for the smallest enclosing circle that contains at least k points in the plane. Note that our algorithm can be easily extended to high dimensions. This algorithm improves over previous results, and it can in some sense be interpreted as an extension of Golin *et al.* [GRSS95] closest pair algorithm to the clustering problem (see also the algorithm by Rabin [Rab76] and the survey of Smid on such algorithms [Smi00]).

Getting similar results for other shape fitting problems, like the minimum radius cylinder in three dimensions, remains elusive. Current approaches for approximating it, in the presence of outliers, essentially reduces to the computation of the shortest vertical segment that stabs at least k hyperplanes. See [SY04] for the details. However, the results of Erickson and Seidel [ES95, Eri99] imply that approximating the shortest vertical segment that stabs $d+1$ hyperplanes takes $\Omega(n^d)$ time, under a reasonable computation model, thus implying that this approach is probably bound to fail if we are interested in a near linear time algorithm.

It would be interesting to figure out which of the shape fitting problems can be approximated in near linear time, in the presence of outliers, and which ones can not. We leave this as an open problem for further research.

Acknowledgments

The authors thank Alon Efrat, Edgar Ramos and Kasturi Varadarajan for helpful discussions on the problems studied in this paper. The authors would also like to thank the anonymous referees for their insightful comments and suggestions.

References

[ABMS98] P. K. Agarwal, M. de Berg, J. Matoušek, and O. Schwarzkopf. *Constructing levels in arrangements and higher order Voronoi diagrams*. *SIAM J. Comput.*, 27: 654–667, 1998.

[AHV04] P. K. Agarwal, S. Har-Peled, and K. R. Varadarajan. *Approximating extent measures of points*. *J. Assoc. Comput. Mach.*, 51(4): 606–635, 2004.

[AST94] P. K. Agarwal, M. Sharir, and S. Toledo. *Applications of parametric searching in geometric optimization*. *J. Algorithms*, 17: 292–318, 1994.

[BE97] M. Bern and D. Eppstein. *Approximation algorithms for geometric problems*. *Approximationg algorithms for NP-Hard problems*. Ed. by D. S. Hochbaum. PWS Publishing Company, 1997, pp. 296–345.

[Cha05] T. M. Chan. *Low-dimensional linear programming with violations*. *SIAM J. Comput.*, 34(4): 879–893, 2005.

[DLSS95] A. Datta, H.-P. Lenhof, C. Schwarz, and M. Smid. *Static and dynamic algorithms for k-point clustering problems. J. Algorithms*, 19: 474–503, 1995.

[EE94] D. Eppstein and J. Erickson. *Iterated nearest neighbors and finding minimal polytopes*. *Discrete Comput. Geom.*, 11: 321–350, 1994.

[Eri99] J. Erickson. *New lower bounds for convex hull problems in odd dimensions*. *SIAM J. Comput.*, 28: 1198–1214, 1999.

[ES95] J. Erickson and R. Seidel. *Better lower bounds on detecting affine and spherical degeneracies*. *Discrete Comput. Geom.*, 13: 41–57, 1995.

[ESZ94] A. Efrat, M. Sharir, and A. Ziv. *Computing the smallest k-enclosing circle and related problems. Comput. Geom. Theory Appl.*, 4: 119–136, 1994.

[GRSS95] M. Golin, R. Raman, C. Schwarz, and M. Smid. *Simple randomized algorithms for closest pair problems. Nordic J. Comput.*, 2: 3–27, 1995.

[HS03] S. Har-Peled and S. Mazumdar. *Fast algorithms for computing the smallest k-enclosing disc. Proc. 11th Annu. Euro. Sympos. Alg.* (ESA)*,* vol. 2832. 278–288, 2003.

[Mat95a] J. Matoušek. *On enclosing k points by a circle. Inform. Process. Lett.*, 53: 217–221, 1995.

[Mat95b] J. Matoušek. *On geometric optimization with few violated constraints*. *Discrete Comput. Geom.*, 14: 365–384, 1995.

[Rab76] M. O. Rabin. *Probabilistic algorithms. Algorithms and Complexity: New Directions and Recent Results.* Ed. by J. F. Traub. Orlando, FL, USA: Academic Press, 1976, pp. 21–39.

[Smi00] M. Smid. *Closest-point problems in computational geometry. Handbook of Computational Geometry.* Ed. by J.-R. Sack and J. Urrutia. Amsterdam, The Netherlands: Elsevier, 2000, pp. 877–935.

[SY04] S. Har-Peled and Y. Wang. *Shape fitting with outliers*. *SIAM J. Comput.*, 33(2): 269–285, 2004.