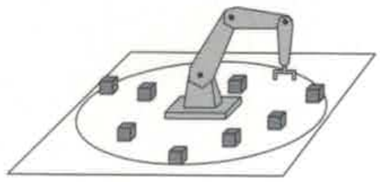
Excerpt from
Computational Geometry — Algorithms and Applications

By de Berg, van Kreveld, Overmars, and Schwarzkopf

4.7* Smallest Enclosing Discs

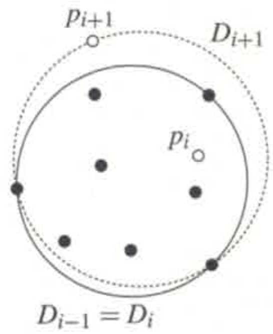
The simple randomized technique we used above turns out to be surprisingly powerful. It can be applied not only to linear programming but to a variety of other optimization problems as well. In this section we shall look at one such problem.



Consider a robot arm whose base is fixed to the work floor. The arm has to pick up items at various points and place them at other points. What would be a good position for the base of the arm? This would be somewhere "in the middle" of the points it must be able to reach. More precisely, a good position is at the center of the smallest disc that encloses all the points. This point minimizes the maximum distance between the base of the arm and any point it has to reach. We arrive at the following problem: given a set P of n points in the plane (the points on the work floor that the arm must be able to reach), find the *smallest enclosing disc* for P, that is, the smallest disc that contains all the points of P. This smallest enclosing disc is unique—see Lemma 4.14(i) below, which is a generalization of this statement.

As in the previous sections, we will give a randomized incremental algorithm for the problem: First we generate a random permutation $p_1,\ldots,p_n$ of the points in P. Let $P_i := \{p_1,\ldots,p_i\}$. We add the points one by one while we maintain D_i, the smallest enclosing disc of P_i.

In the case of linear programming, there was a nice fact that helped us to maintain the optimal vertex: when the current optimal vertex is contained in the next half-plane then it does not change, and otherwise the new optimal vertex lies on the boundary of the half-plane. Is a similar statement true for smallest enclosing discs? The answer is yes:



Lemma 4.13 *Let $2 < i < n$, and let P_i and D_i be defined as above. Then we have*
(i) If $p_i \in D_{i-1}$, then $D_i = D_{i-1}$.
(ii) If $p_i \notin D_{i-1}$, then p_i lies on the boundary of D_i.

We shall prove this lemma later, after we have seen how we can use it to design a randomized incremental algorithm that is quite similar to the linear programming algorithm.

```
Algorithm MINIDISC($P$)
Input. A set $P$ of $n$ points in the plane.
Output. The smallest enclosing disc for $P$.
1.  Compute a random permutation $p_1, \dots, p_n$ of $P$.
2.  Let $D_2$ be the smallest enclosing disc for $\{p_1, p_2\}$.
3.  for $i \leftarrow 3$ to $n$
4.      do if $p_i \in D_{i-1}$
5.          then $D_i \leftarrow D_{i-1}$
6.          else $D_i \leftarrow$ MINIDISCWITHPOINT($\{p_1, \dots, p_{i-1}\}, p_i$)
7.  return $D_n$
```

The critical step occurs when $p_i \notin D_{i-1}$. We need a subroutine that finds the smallest disc enclosing P_i, using the knowledge that p_i must lie on the boundary of that disc. How do we implement this routine? Let $q := p_i$. We use the same framework once more: we add the points of P_{i-1} in random order, and maintain the smallest enclosing disc of $P_{i-1} \cup \{q\}$ under the extra constraint that it should have q on its boundary. The addition of a point p_j will be facilitated by the following fact: when p_j is contained in the currently smallest enclosing disc then this disc remains the same, and otherwise it must have p_j on its boundary. So in the latter case, the disc has both q and p_j and its boundary. We get the following subroutine.

```
MINIDISCWITHPOINT($P$,$q$)
Input. A set $P$ of $n$ points in the plane, and a point $q$ such that there exists an
    enclosing disc for $P$ with $q$ on its boundary.
Output. The smallest enclosing disc for $P$ with $q$ on its boundary.
1.  Compute a random permutation $p_1, \dots, p_n$ of $P$.
2.  Let $D_1$ be the smallest disc with $q$ and $p_1$ on its boundary.
3.  for $j \leftarrow 2$ to $n$
4.      do if $p_j \in D_{j-1}$
5.          then $D_j \leftarrow D_{j-1}$
6.          else $D_j \leftarrow$ MINIDISCWITH2POINTS($\{p_1, \dots, p_{j-1}\}, p_j, q$)
7.  return $D_n$
```

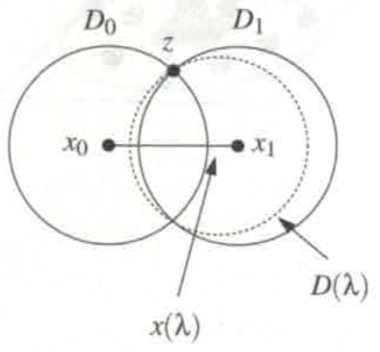
How do we find the smallest enclosing disc for a set under the restriction that two given points q_1 and q_2 are on its boundary? We simply apply the same approach one more time. Thus we add the points in random order and maintain the optimal disc; when the point p_k we add is inside the current disc we don't have to do anything, and when p_k is not inside the current disc it must be on the boundary of the new disc. In the latter case we have three points on the disc boundary: q_1, q_2, and p_k. This means there is only one disc left: the unique disc with q_1, q_2, and p_k on its boundary. This following routine describes this in more detail.

```
MINIDISCWITH2POINTS($P, q_1, q_2$)
Input. A set $P$ of $n$ points in the plane, and two points $q_1$ and $q_2$ such that there
   exists an enclosing disc for $P$ with $q_1$ and $q_2$ on its boundary.
Output. The smallest enclosing disc for $P$ with $q_1$ and $q_2$ on its boundary.
1.  Let $D_0$ be the smallest disc with $q_1$ and $q_2$ on its boundary.
2.  for $k \leftarrow 1$ to $n$
3.      do if $p_k \in D_{k-1}$
4.           then $D_k \leftarrow D_{k-1}$
5.           else $D_k \leftarrow$ the disc with $q_1$, $q_2$, and $p_k$ on its boundary
6.  return $D_n$
```

This finally completes the algorithm for computing the smallest enclosing disc of a set of points. Before we analyze it, we must validate its correctness by proving some facts that we used in the algorithms. For instance, we used the fact that when we added a new point and this point was outside the current optimal disc, then the new optimal disc must have this point on its boundary.

Lemma 4.14 *Let P be a set of points in the plane, let R be a possibly empty set of points with $P \cap R = \emptyset$, and let $p \in P$. Then the following holds:*
(i) If there is a disc that encloses P and has all points of R on its boundary, then the smallest such disc is unique. We denote it by $md(P,R)$.
(ii) If $p \in md(P \setminus \{p\}, R)$, then $md(P,R) = md(P \setminus \{p\}, R)$.
(iii) If $p \notin md(P \setminus \{p\}, R)$, then $md(P,R) = md(P \setminus \{p\}, R \cup \{p\})$.

Proof. (i) Assume that there are two distinct enclosing discs D_0 and D_1 with centers x_0 and x_1, respectively, and with the same radius. Clearly, all points of P must lie in the intersection $D_0 \cap D_1$. We define a continuous family $\{D(\lambda) \mid 0 \leqslant \lambda \leqslant 1\}$ of discs as follows. Let z be an intersection point of ∂D_0 and ∂D_1, the boundaries of D_0 and D_1. The center of $D(\lambda)$ is the point $x(\lambda) := (1-\lambda)x_0 + \lambda x_1$, and the radius of $D(\lambda)$ is $r(\lambda) := d(x(\lambda), z)$. We have $D_0 \cap D_1 \subset D(\lambda)$ for all λ with $0 \leqslant \lambda \leqslant 1$ and, in particular, for $\lambda = 1/2$. Hence, since both D_0 and D_1 enclose all points of P, so must $D(1/2)$. Moreover, $\partial D(1/2)$ passes through the intersection points of ∂D_0 and ∂D_1. Because $R \subset \partial D_0 \cap \partial D_1$, this implies that $R \subset \partial D(1/2)$. In other words, $D(1/2)$ is an enclosing disc for P with R on its boundary. But the radius of $D(1/2)$ is strictly less than the radii of D_0 and D_1. So whenever there are two distinct enclosing discs of the same radius with R on their boundary, then there is a smaller enclosing disc with R on its boundary. Hence, the smallest enclosing disc $md(P,R)$ is unique.



(ii) Let $D := md(P \setminus \{p\}, R)$. If $p \in D$, then D contains P and has R on its boundary. There cannot be any smaller disc containing P with R on its boundary, because such a disc would also be a containing disc for $P \setminus \{p\}$ with R on its boundary, contradicting the definition of D. It follows that $D = md(P,R)$.

(iii) Let $D_0 := md(P \setminus \{p\}, R)$ and let $D_1 := md(P,R)$. Consider the family $D(\lambda)$ of discs defined above. Note that $D(0) = D_0$ and $D(1) = D_1$, so the family defines a continous deformation of D_0 to D_1. By assumption we have $p \notin D_0$. We also have $p \in D_1$, so by continuity there must be some $0 < \lambda^* \leqslant 1$ such that p lies on the boundary of $D(\lambda^*)$. As in the proof of (i), we have $P \subset D(\lambda^*)$ and $R \subset \partial D(\lambda^*)$. Since the radius of any $D(\lambda)$ with $0 < \lambda < 1$ is strictly less than the radius of D_1, and D_1 is by definition the smallest enclosing disc for P, we must have $\lambda^* = 1$. In other words, D_1 has p on its boundary.

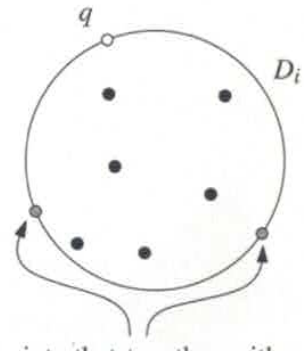
□

Lemma 4.14 implies that MINIDISC correctly computes the smallest enclosing disc of a set of points. The analysis of the running time is given in the proof of the following theorem.

Theorem 4.15 *The smallest enclosing disc for a set of n points in the plane can be computed in $O(n)$ expected time using worst-case linear storage.*

Proof. MINIDISCWITH2POINTS runs in $O(n)$ time because every iteration of the loop takes constant time, and it uses linear storage. MINIDISCWITHPOINT and MINIDISC also need linear storage, so what remains is to analyze their expected running time.

The running time of MINIDISCWITHPOINT is $O(n)$ as long as we don't count the time spent in calls to MINIDISCWITH2POINTS. What is the probability of having to make such a call? Again we use backwards analysis to bound this probability: Fix a subset $\{p_1, \dots, p_i\}$, and let D_i be the smallest disc enclosing $\{p_1, \dots, p_i\}$ and having q on its boundary. Imagine that we remove one of the points $\{p_1, \dots, p_i\}$. When does the smallest enclosing circle change? That happens only when we remove one of the three points on the boundary. (When there are more points on the boundary, then the smallest disc cannot change after removing one point.) One of the points on the boundary is q, so there are at most two points that cause the smallest enclosing circle to shrink. The probability that p_i is one of those points is $2/i$. So we can bound the total expected running time of MINIDISCWITHPOINT by



points that together with q define D_i

$$O(n) + \sum_{i=2}^{n} O(i)\frac{2}{i} = O(n).$$

Applying the same argument once more, we find that the expected running time of MINIDISC is $O(n)$ as well. □

Algorithm MINIDISC can be improved in various ways. First of all, it is not necessary to use a fresh random permutation in every instance of subroutine

MINIDISCWITHPOINT. Instead, one can compute a permutation once, at the start of MINIDISC, and pass the permutation to MINIDISCWITHPOINT. Furthermore, instead of writing three different routines, one could write a single algorithm MINIDISCWITHPOINTS(P,R) that computes $md(P,R)$ as defined in Lemma 4.14.

4.8 Notes and Comments

It is only recently that the geometric computations needed in computer aided manufacturing have received consideration from computational geometry. We have only scratched the surface of the area of manufacturing processes; Bose and Toussaint [52] give an extensive overview.

The computation of the common intersection of half-planes is an old and well-studied problem. As we will explain in Chapter 11, the problem is dual to the computation of the convex hull of points in the plane. Both problems have a long history in the field, and Preparata and Shamos [289] already list a number of solutions. More information on the computation of 2-dimensional convex hulls can be found in the notes and comments of Chapter 1.

Computing the common intersection of half-spaces, which can be done in $O(n \log n)$ time in the plane and in 3-dimensional space, becomes a more computationally demanding problem when the dimension increases. The reason is that the number of (lower-dimensional) faces of the convex polytope formed as the common intersection can be as large as $\Theta(n^{\lfloor d/2 \rfloor})$ [131]. So if the only goal is to find a feasible point, computing the common intersection explicitly soon becomes an unattractive approach.

Linear programming is one of the basic problems in numerical analysis and combinatorial optimization. It goes beyond the scope of this chapter to survey this literature, and we restrict ourselves to mentioning the simplex algorithm and its variants [117], and the polynomial-time solutions of Khachiyan [202] and Karmarkar [197]. More information on linear programming can be found in books by Chvátal [106] and Schrijver [305].

Linear programming as a problem in computational geometry was first considered by Megiddo [242], who showed that the problem of testing whether the intersection of half-spaces is empty is strictly simpler than the computation of the intersection. He gave the first deterministic algorithm for linear programming whose running time is of the form $O(C_d n)$, where C_d is a factor depending on the dimension only. His algorithm is linear in n for any fixed dimension. The factor C_d in his algorithm is 2^{2^d}. This was later improved to 3^{d^2} [108, 127]. More recently, a number of simpler and more practical randomized algorithms have been given [110, 312, 317]. There are a number of randomized algorithms whose running time is subexponential, but still not polynomial in the dimension [192, 236]. Finding a strongly polynomial algorithm, that is of combinatorial polynomial complexity, for linear programming is one of the major open problems in the area.

The simple randomized incremental algorithm for two and higher dimen-