

平面点集的最小包围圆

1、 问题背景

考察固定在工作平台上的一直机械手，要捡起散落在不同位置的多个零件，并送到别的地方。那么，这只机械手的底座应该选在哪里呢？根据直觉，应该选在机械手需够着的那些位置的“中心”。准确地讲，也就是包围这些点的那个最小圆的圆心——该位置的好处是，可使机械手的底座到它需要够着的那些点的最大距离最小化。于是可得如下问题：给定由平面上 n 个点所组成的一个集合 P （对应于机械手需要够着的工作平台的那些位置），试找出 P 的最小包围圆（smallest enclosing disc）——亦即，包含 P 中所有点、半径最小的那个圆。这个最小包围圆必然是唯一的。

2、 算法及原理

算法介绍：我们本次算法的设计是基于这样一个简单直观的性质：在既定的给定点条件下，如果引入一张新的半平面，只要此前的最优解顶点（即唯一确定最小包围圆的几个关键顶点）能够包含于其中，则不必对此最优解进行修改，亦即此亦为新点集的最优解；否则，新的最优解顶点必然位于这个新的半空间的边界上。

定理可以通过反证法证明。

于是，基于此性质，我们便可得到一个类似于线性规划算法的随机增量式算法。定义 D_i 为相对于 p_i 的最小包围圆。此算法实现的关键在于对于 $p_i \notin D_{i-1}$ 时的处理。显然，如果 $p_i \in D_{i-1}$ ，则 $D_i = D_{i-1}$ ；否则，需要对 D_i 另外更新。而且， D_i 的组成必然包含了 p_i ；因此，此种情况下的最小包围圆是过 p_i 点且覆盖点集 $\{p_1, p_2, p_3, \dots, p_{i-1}\}$ 的最小包围圆。则仿照上述处理的思路， $D_i = \{p_1, p_i\}$ ，逐个判断点集 $\{p_2, p_3, \dots, p_{i-1}\}$ ，如果存在 $p_j \notin D_i$ ，则 $D_i = \{p_j, p_i\}$ 。同时，再依次对点集 $\{p_1, p_2, p_3, \dots, p_{j-1}\}$ 判断是否满足 $p_k \in D_i$ ，若有不满足，则 $D_i = \{p_k, p_j, p_i\}$ 。由于，三点唯一地确定一个圆，故而，只需在此基础上判断其他的点是否位于此包围圆内，不停地更新 p_k 。当最内层循环完成时，退出循环，转而更新 p_j ；当次内层循环结束时，退出循环，更新 p_i 。当 $i=n$ 时，表明对所有的顶点均已处理过，此时的 D_n 即表示覆盖了给定 n 个点的最小包围圆。

算法 MINIDISC (P)

输入：由平面上 n 个点组成的一个集合 P

输出： P 的最小包围圆

1. 令 D_2 为对应于 $\{p_1, p_2\}$ 的最小包围圆
2. for $i \leftarrow 3$ to n
3. do if $p_i \in D_{i-1}$
4. then $D_i \leftarrow D_{i-1}$
5. else $D_i \leftarrow \text{MINIDISCWITHPOINT}(\{p_1, p_2, p_3, \dots, p_{i-1}\}, p_i)$
6. return D_n

算法 MINIDISCWITHPOINT (P, q)

输入：由平面上 n 个点构成的一个集合 P ，以及另外一个点 q

输出：在满足“边界穿过 q ”的前提下， P 的最小包围圆

1. 令 D_1 为对应于 $\{ p_1, q \}$ 的最小包围圆
2. for $j \leftarrow 2$ to n
3. do if $p_j \in D_{j-1}$
4. then $D_j \leftarrow D_{j-1}$
5. else $D_j \leftarrow \text{MINIDISCWITH2POINT}(\{ p_1, p_2, p_3, \dots, p_{j-1} \}, p_j, p_1)$
6. return D_n

算法 MINIDISCWITH2POINT (P, q_1, q_2)

输入: 由平面上 n 个点构成的一个集合 P , 以及另外两个点 q_1, q_2

输出: 在满足“边界穿过 q_1, q_2 ”的前提下, P 的最小包围圆

1. 令 D_0 为对应于 $\{ q_1, q_2 \}$ 的最小包围圆
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 $D_k \leftarrow q_1, q_2$ 和 p_k 确定的圆
6. return D_n

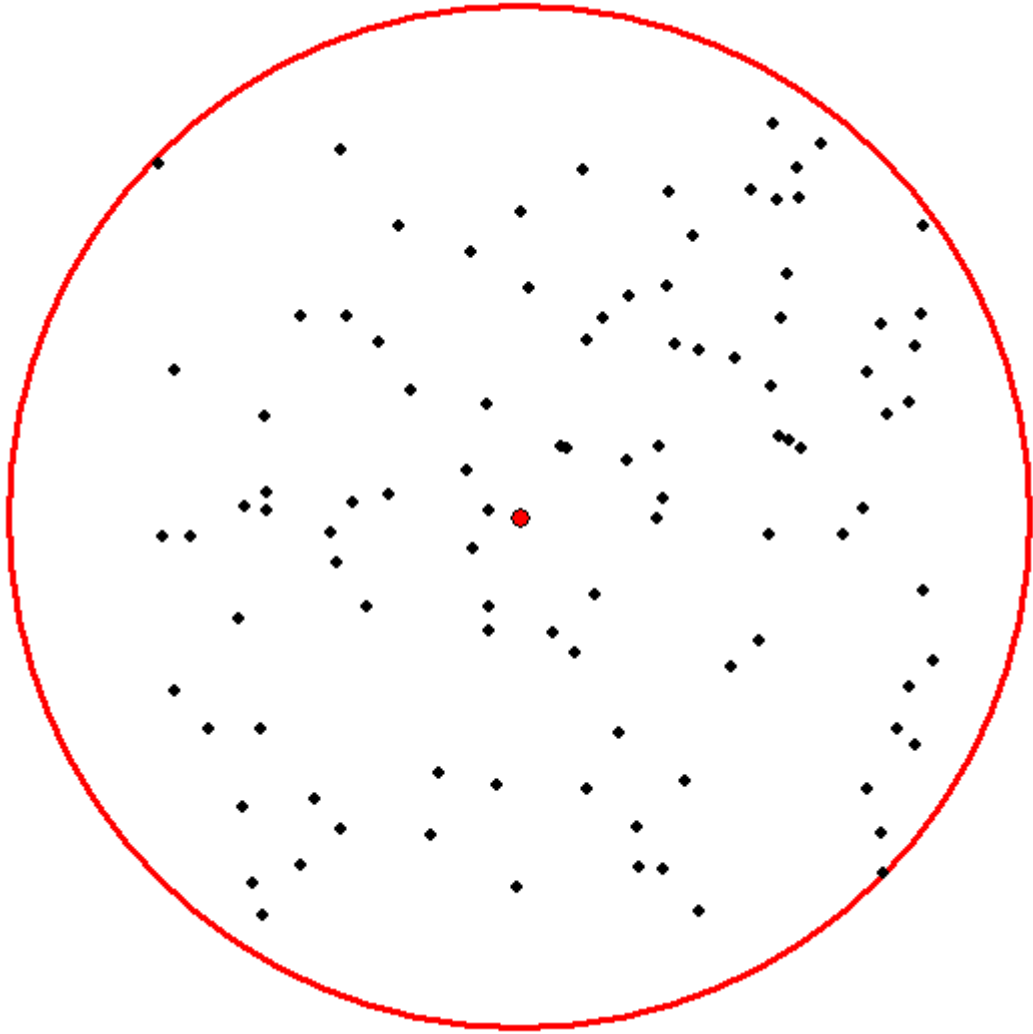
时间复杂度: 此算法对于任意给定的 n 个点, 可以再 $O(n)$ 的期望运行时间内计算出来。

算法 MINIDISCWITH2POINT 中的每一轮迭代循环只需要常数时间, 因此其运行时间为 $O(n)$; 另外, 它只需要线性的空间。至于算法 MINIDISCWITHPOINT 和 MINIDISC, 他们也同样只需要线性的存储空间。对于它们的期望运行时间分析如下: 对于算法 MINIDISCWITHPOINT, 只要不计入其调用 MINIDISCWITH2POINT 的时间, 其余计算所需要的时间为 $O(n)$ 。依据概率论分析可以求其期望运行时间的上界 $O(n) +$

2、 遇到的问题及解决对策

实验之初, 我们曾提出一种比较直观的算法, 即从平面上的任意三点出发, 求其最小包围圆,; 再依次判断之外的点, 看其相对位置是位于圆内 (包括在圆上) 还是圆外。若在圆内, 则最小包围圆不变, 再判断下个点; 否则, 求包围此四点的最小包围圆。逐个迭代, 直至完全遍历。显然, 该算法的时间复杂度比较大, 达到 $O(n^4)$, 在实际应用中是不可取的。

4、 测试结果



红色的点为求出的圆心位置显示。

算法效率说明

数据规模	算法执行时间 (ms)
50	0
500	16
5000	110
50000	359

5、程序说明

1. 在空白区域可以直接点击鼠标逐个输入点数据。(目前支持最大 65536 个点的数据)，直接计算出最小覆盖圆的圆心，半径

2. 菜单栏“数据”-->“清空数据”： 将已有的数据信息删除，以便完成新的输入操作。

3. 菜单栏“数据”-->“随机数”： 输入需要的随机点个数，产生随机点。（目前支持最大 128 个点的数据）
4. 菜单栏“数据”-->“保存样例”： 保存当前输入点的信息，存储为 txt 文件。最后一行的结果为圆心的坐标，圆的半径
5. 菜单栏“数据”-->“选择样例”： 根据选择的 txt 文件，批量生成点的信息数据
6. 状态栏上给出了算法的执行时间

6、参考文献

1. smallest enclosing disks balls and ellipsoids by EMO Welzl
2. 计算几何-算法与应用（第二版）