平面点集的最小包围圆算法设计及实现

王立（P1016017） 张涵初（P1016015） 张超（P1016005）

华北计算技术研究所

一、问题背景

欲在一平面区域内建立一个固定的无线信号收发中转基站，平面区域内有若干个固定的无线通信用户（可抽象为平面内一点）。各无线通信用户之间若要实现互联通信，均需要无线通信用户先与中转基站建立无线链接，再通过中转基站链接区域内任一点。那么这个无线信号中转基站应该选在哪里呢？根据直觉，应该选在中转基站射频作用距离范围的“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，中转基站与它射频作用范围内的点实现通信时，射频发射机的发射功率即功耗达到最小化（无线射频通信中，作用距离越短，射频发射机功耗越小）。于是可抽象出如下问题：给定由平面上 n 个点所组成的一个集合 P（对应于中转基站射频作用范围覆盖的区域内的通信用户），试找出 P 的最小包围圆（smallest enclosing disc），即包含 P 中所有点并且半径最小的那个圆。可以证明，这个最小包围圆有且仅有一个。

二、实验内容

本实验主要完成了平面上随机点集P的生成和关于P的最小包围圆的计算。在生成平面随机点集时，通过伪随机函数rand()函数实现了在平面矩形区域、平面圆形区域以及平面环形区域内的均匀随机分布。在计算最小包围圆的过程中，使用并改进了一种比较高效的线性算法，在O(n)的时间内计算出了覆盖平面点集P的最小圆。

生成平面随机点集P时，我们将点的分布区域设置为平面矩形、圆形和扇形等，并将上述算法分别作用到不同分布区域的平面点集P，对比分析了其各自时间复杂度。

三、算法的理论描述

在我们熟悉的二维平面的线性规划问题中，在目标函数和给定的初始边界条件共同确定出最优顶点P时，若引进一张半平面S_1，顶点P落在S_1所包围的区域内，则当前的最优点可以保存前一次所求出的最优顶点P而不必修改。将此思路应用于最小外接圆算法：对于由平面点集P内的多个点p_1，p_2，……，p_i（相当于i个初始边界条件）确定出一个最小包围圆C_i，如果引入一张新的半平面S_{i+1}，此半平面为以C_i的圆心o_i为圆心，以平面上不包含于P的一点p_{i+1}与C_i的圆心o_i的距离$|p_{i+1}o_i|$为半径。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）能够包含于半平面S_{i+1}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。其形式如图 1 所示：

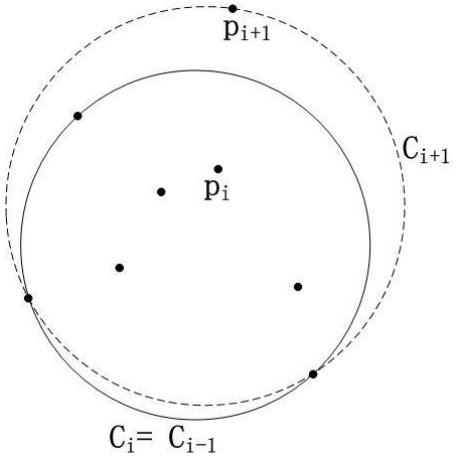


图 1——若$p_i \in C_{i-1}$，则$C_i = C_{i-1}$，若$p_{i+1} \notin C_i$，则C_{i+1}更新

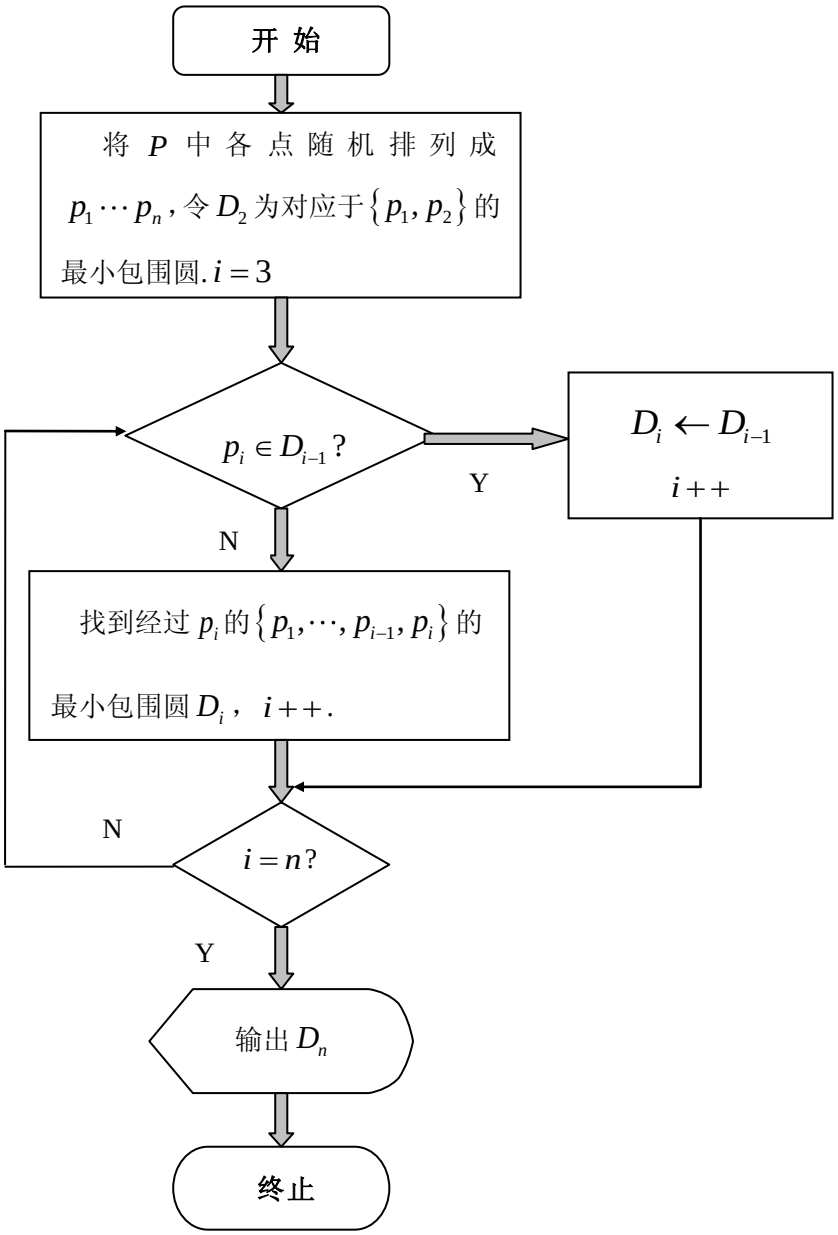
上述性质为一几何引理，它构成了本文及本次实验的理论基石，本文所讲述的算法也将围绕该引理展开。

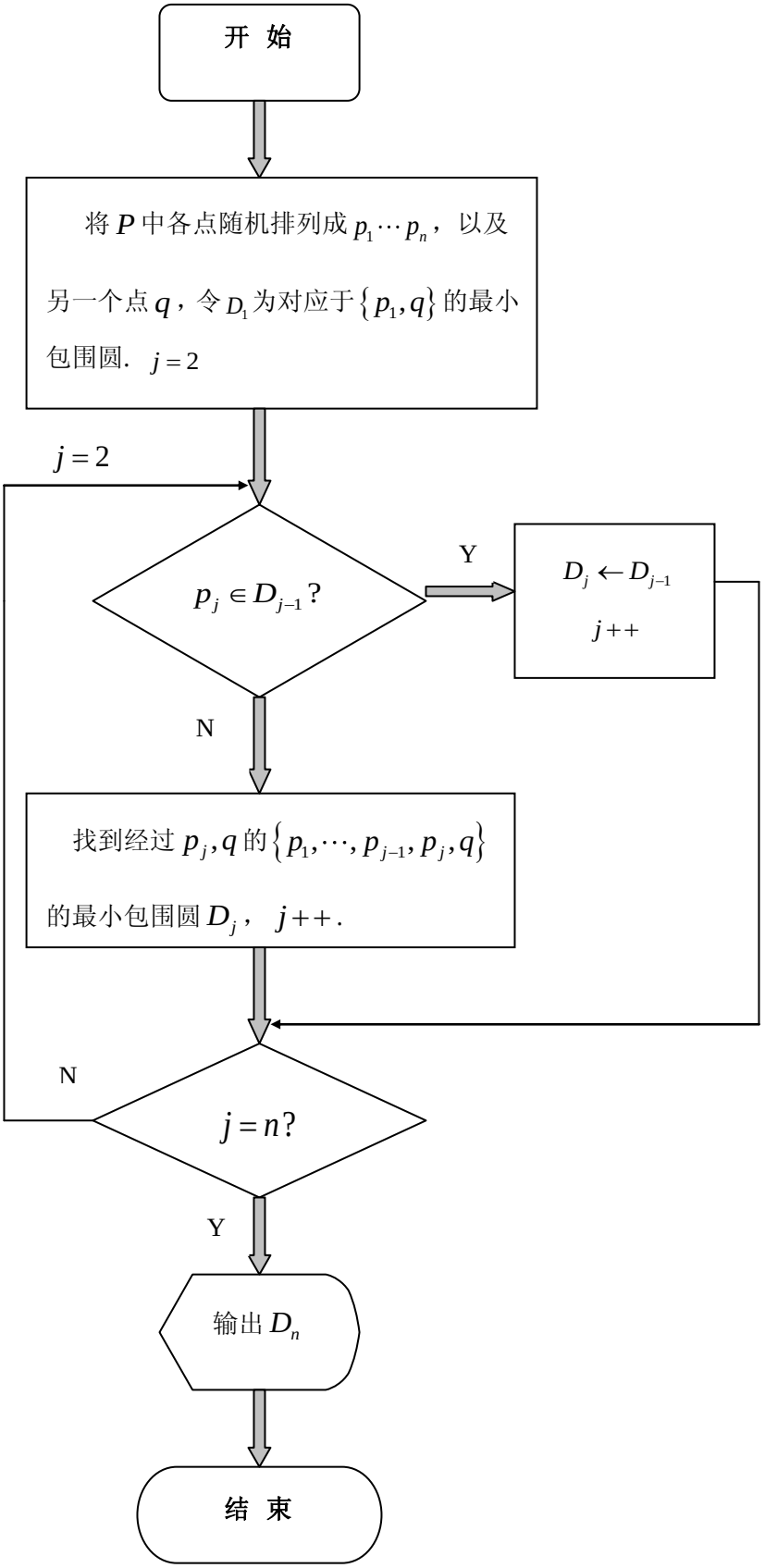
仿照线性规划的随机增量式算法，我们可以得到期望在线性时间内的完成的最小包围圆算法，将此算法命名为 MCC 算法（Minimum Circumscribed Circle）。定义圆C_i为相对于平面点集P的最小包围圆。根据上述引理，可知当P中新引入一点p_i时，若p_{i+1}包含于圆C_i内，则维护圆C_i，作为此时P的最小包围圆，即$C_{i+1} = C_i$。而此算法实现的关键在于对于$p_{i+1} \notin C_i$时的处理。显然，当$p_{i+1} \notin C_i$，则需要对C_i进行更新。而且，圆C_{i+1}必然经过p_{i+1}。因此，此种情况下的最小包围圆是过p_{i+1}点且覆盖点集P $= \{p_1, p_2, \ldots\ldots, p_i\}$的最小包围圆。则仿照上述处理的思路，初始化圆$C_i$的修改值，令其为圆$C_i$‘，此圆通过点$p_1$和$p_{i+1}$，进而逐个扫描判断点集$\{p_2, p_3, \ldots\ldots, p_i\}$。当$p_k \in C_i$’，则维护$C_i$’不变，如果存在$p_k \notin C_i$’，则再次修改圆$C_i$’的值为圆$C_i$“，此时圆$C_i$“通过点$p_k, p_{i+1}$。接着，再依次对点集$\{p_1, p_2, \ldots\ldots, p_{i-1}\}$进行逐个扫描，判断其是否满足$p_l \in C_i$”，若存在$p_l \notin C_i$“，则$C_i$”即由点$p_k, p_l, p_{i+1}$三点确定。

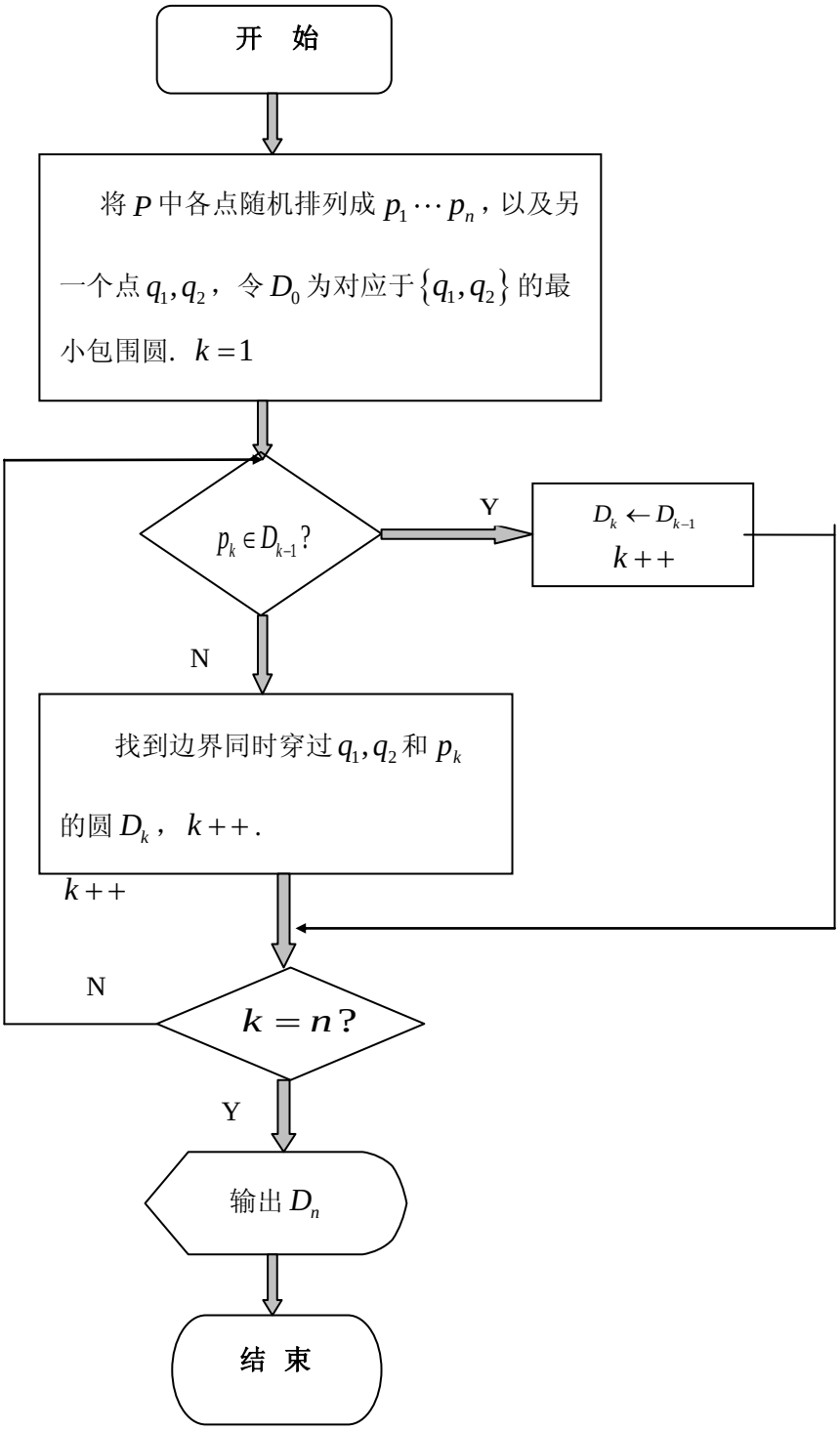
由于三点唯一地确定一个圆，故而，只需在此基础上判断其他的点是否位于此包围圆内。算法由两层递归调用，三层循环完成。即首先不停地更新p_l，当最内层循环完成时，退出循环，转而更新p_k；当次内层循环结束时，退出循环，更新p_i。当i = n时，表明对所有的顶点均已处理过 ，此时的C_n即为覆盖了给定平面上 n 个不同点的最小包围圆。算法流程图如图 2 所示。

四、算法的实现

根据前文算法的理论描述，设计出算法流程图如下：




开 始
将 P 中各点随机排列成 $p_1 \cdots p_n$，以及另一个点 q，令 D_1 为对应于 $\{p_1, q\}$ 的最小包围圆. $j=2$
$j=2$
$p_j \in D_{j-1}$?
Y
$D_j \leftarrow D_{j-1}$
$j{+}{+}$
N
找到经过 p_j, q 的 $\{p_1, \cdots, p_{j-1}, p_j, q\}$ 的最小包围圆 D_j， $j{+}{+}$.
N
$j=n$?
Y
输出 D_n
结 束



用伪代码描述如下：

Function MiniDisc（P）

Input：由平面上 n 个点组成的一个集合P $=\{p_1, p_2, \ldots\ldots, p_i\}$

Output：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← 3 to n

3. do if $p_i \in D_{i-1}$

4. then $D_i \leftarrow D_{i-1}$

5. else $D_i \leftarrow$ MiniDiscWithPoint{{ p_1 ， p_2 ， p_3 ……p_{i-1}}， p_i}

6. return D_n

Function MiniDiscWithPoint（P，q）

Input：由平面上 n 个点构成的一个集合 P，以及另外一个点 q

Output：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 $D_j \leftarrow$ MiniDiscWith2Points {{ p_1 ， p_2 ， p_3 ……p_{j-1}}， p_j， p_i}

6. return D_n

Function MiniDiscWith2Points（P，q_1，q_2）

Input：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

Output：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

在算法实现过程中，使用结构体 m_circle 维护动态维护当前最小包围圆C_i，平面点集 $P = \{p_1, p_2, \ldots\ldots, p_i\}$则保存在点队列 m_pointsArray 中，其数据结构如图 3 所示。CShape 类为基类，CGlPoint 类和 CGICircle 类分别为其继承类，同时也是点对象和圆对象的数据结构。多个点对象组成点队列，点队列用 C++ STL 中的 Vector 进行组织。在程序中的表示分别为点队列 m_pointsArray 和圆对象 m_circle。

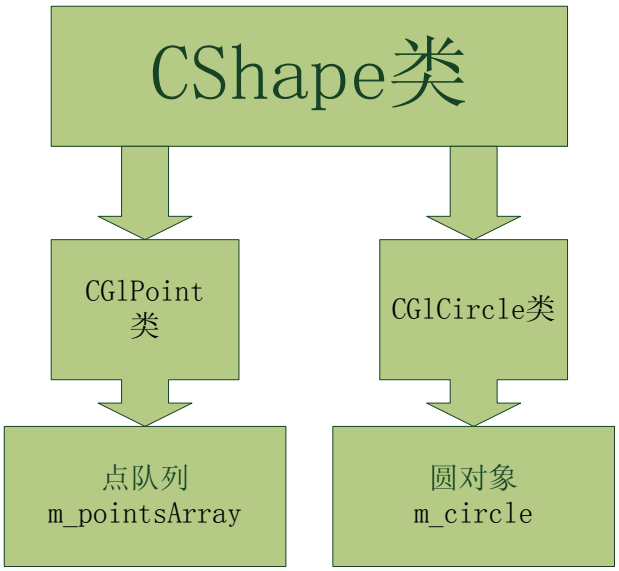


图 3——点队列和圆对象的数据结构示意图

五、算法分析：

1、时间复杂度分析

前文已提及，平面上任意一组共n个点的最小包围圆，可以再O（n）的期望运行时间内计算出来，并且为此需要的空间在最坏的情况下不会超过线性规模。现给出具体分析：

函数MiniDiscWith2Points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，它只需要线性空间。同理，函数MiniDisc和MiniDiscWithPoint也只需要线性的空间。故需要对MiniDisc和MiniDiscWithPoint的期望运行时间进一步分析。

对于函数MiniDiscWithPoint，只要不计入其调用函数MiniDiscWith2Points的时间，其余部分需要的时间为O（n）。采用后向分析方法，假设平面点集P = $\{p_1, p_2, \ldots\ldots, p_i\}$已固定，令$C_{i+1}$为覆盖点集P、其边界穿过不包含于P的点$p_{i+1}$的最小圆。然后，删去P中某个点，分析在哪些情况下圆C_{i+1}会发生变化。经过分析可知，只有当删去落于圆上的三个点之中的一个，圆C_{i+1}才会发生变化。如图 4 所示：

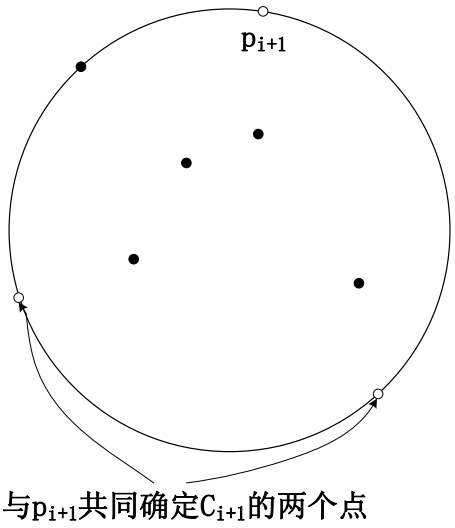


图 4——除去p_{i+1}外，至多还有两个点的删除会导致最小包围圆缩小

当圆上的点数超过三个时，即使删掉圆上的点，圆C_{i+1}也不会发生改变。如图所示。又因为p_{i+1}不属于P，故平面点集P中能使C_{i+1}缩小的点最多有两个。再正向分析，可知每新引入这两个点中的一个，当前维护的最小包围圆均会增大，每次增大都要调用一次函数MiniDiscWith2Points，能够调用函数MiniDiscWith2Points的概率为$2/i$，则函数MiniDiscWithPoint的期望运行时间上界为：

$$O(n)+\sum_{i=2}^{n} O(i)\cdot 2/i = O(n)$$

同理可以分析得到，函数MiniDisc的期望运行时间也是$O(n)$。故算法 MCC 的期望运行时间以$O(n)$为上界。

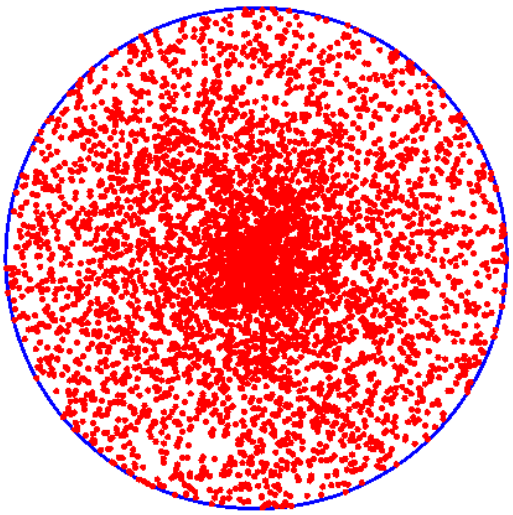
2、算法实现及效率分析

算法程序的开发环境主要包括两个方面：硬件环境和软件环境。本程序开发的硬件和软件环境如表 3：

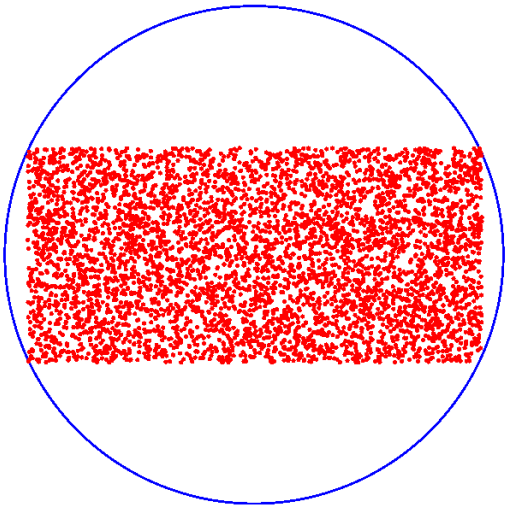
表 3——开发环境软硬件配置

硬件环境	
CPU	Intel Core2 T8100 2.10GHz*2
内存	2G
硬盘	160G
显卡	NVIDIA GeForce 9500M GS
软件环境	
操作系统	Windows 7 Pro
集成开发环境	Visual Studio 2008 Pro
开发语言	C++

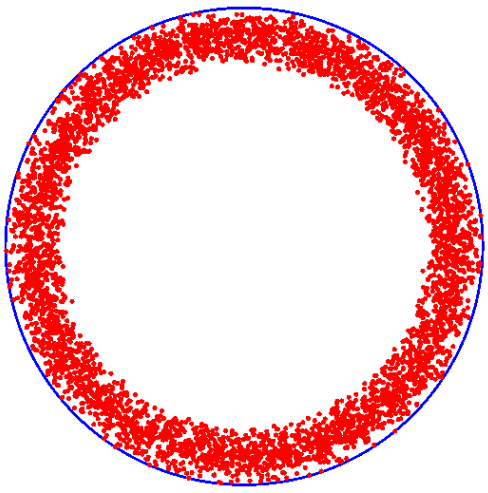
我们分别选取不同形状的分布区域进行随机点生成，计算出在圆形、矩形、环形等区域内随机分布点集的最小包围圆，并测试其计算时间，以验证其期望线性的时间复杂度。测试结果见图 5 和表 1。



5.1 圆形分布区域



5.2 矩形分布区域



5.3 环形分布区域

图 5—5000 点随机均匀在不同区域时分布程序运行结果

表 1—不同分布区域下均匀随机分布点数生成最小包围圆所用时间（单位：ms）

分布区域 / 点数规模	矩形分布	圆形分布	环形分布
5000	21	29	25
10000	40	46	42
50000	93	105	149
10000	191	185	257
500000	799	703	1089
1000000	1859	1573	2218

从表 1 中可以看出，对于不同的分布区域，算法效率略有不同，但各自基本服从线性递增的规律。

3、平面点集分布不同时算法的效率——对期望线性时间复杂度的验证对比

由上文分析得知，算法 MCC 时间复杂度以$O(n)$为期望。但是，根据直观判断，在上层函数中调用下层函数的次数愈少，算法的效率就越高。前文给出的 MCC 算法复杂度分析是基于随机均匀分布。而对于不同分布的平面点集，使用 MCC 算法将可能导致算法的时间复杂度不同。验证如下：

在生成平面均匀分布随机点集P时，我们使用 VC 中的伪随机函数rand()函数，产生两组零点周围的均匀随机分布的序列，分别作为P中的点的x，y坐标。产生的矩形区域和圆形区域的点集分布如图 6 所示：

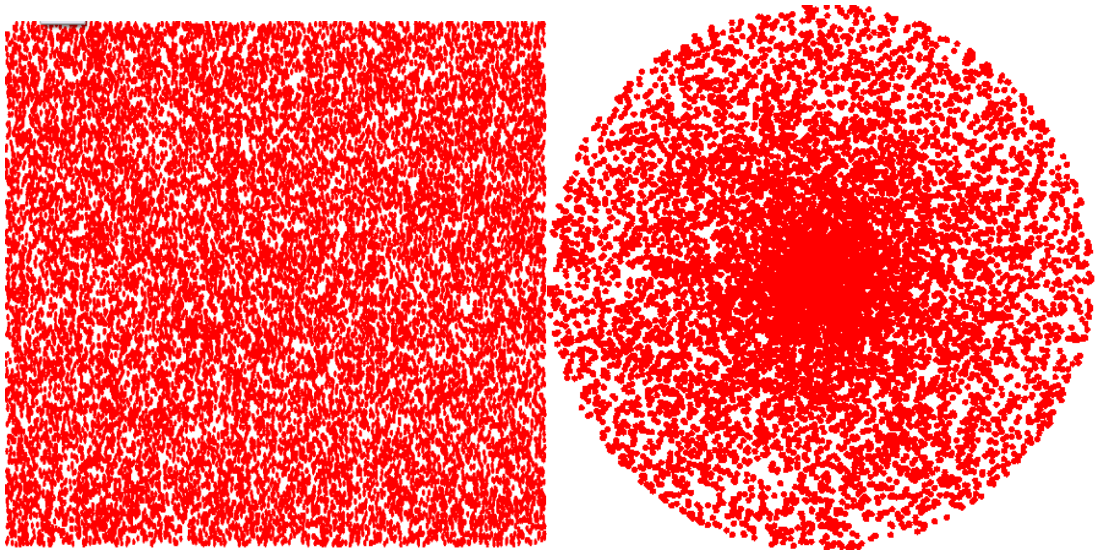


图 6—平面点集的随机均匀分布

从图中可以看出，点在所在区域分布均匀。由于高斯分布为连续性变量，不利于描述离散点的分布，我们考虑使用两个伪随机函数做差，即进行|rand（） – rand（）|运算，以生成两组随机序列，满足在 0 周围的密集化，在距离 0 较远的范围内稀疏化，将这两组序列作为一组点的x，y坐标，即可生成一种中间分布较密集，边缘分布较稀疏的类高斯分布的平面点集P′，其分布如图所示：

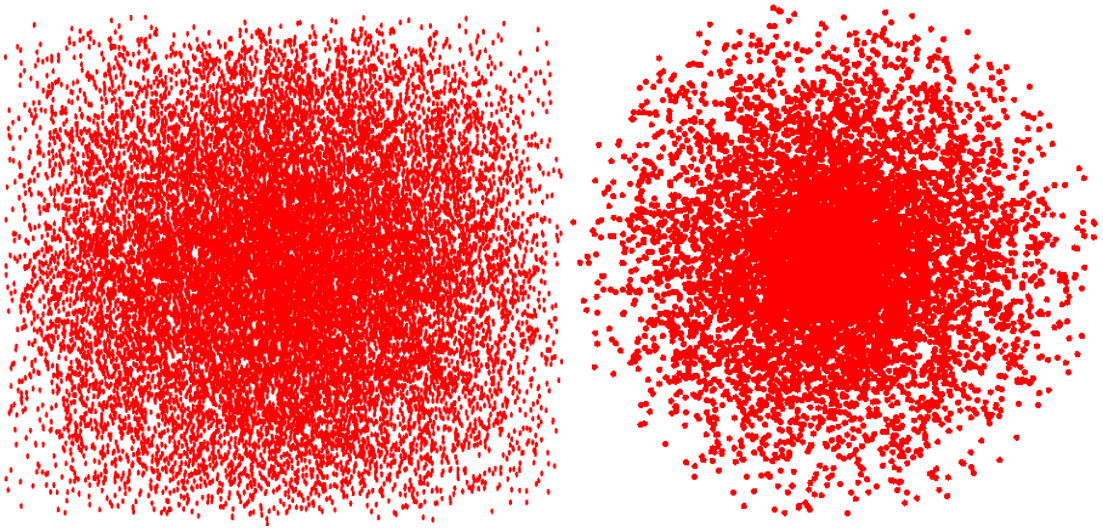


图 7—平面点集的类高斯随机分布

表 2—平面点集在不同分布形式下生成最小包围圆所用时间对比（单位：ms）

分布方式	点数规模（个）	矩形分布（ms）	圆形分布（ms）
类高斯分布	500000	2147	2122
	1000000	2583	3500
均匀分布	500000	799	703
	1000000	1859	1573

从表 2 可以看出，均匀分布下，MCC 算法运行效率较高，而对于中间密集、边沿稀疏的类高斯分布，其时间复杂度明显较高。

六、程序使用说明

本程序用 VS2008 开发、编译、调试完成，程序界面友好美观，使用方便，实现了本次大作业选题的要求，算法时间复杂度较好，运算速度较快，程序吞吐量较大，可以在线性时间内计算百万个随机生成点的最小包围圆的计算，程序使用说明如下：

1、随机点生成

单击菜单选项“测试->生成随机点”，在弹出的对话框中填写随机点数量、随机点分布区域形状（包括矩形、圆形、环形）、分布区域的参数等，再单击“添加”按钮，即可完成随机点集生成。也可通过单击工具栏选项实现。

也可手动添加点，可点集菜单选项“操作->添加点”，然后再程序窗口空白处点击鼠标，所在位置随即产生一个点。可通过单击工具栏选项实现。

2、最小包围圆计算

平面点集生成好以后，单击菜单选项“测试->计算最小包围圆”，程序窗口即可显示包围平面点集的最小包围圆。可通过单击工具栏选项实现。此时若继续添加点，最小包围圆将实时更新。

3、计时功能

平面点集生成好以后，单击菜单选项“测试->计时”计时选项即被对勾选中。再单击菜单选项“测试->计算最小包围圆”，程序窗口即可显示包围平面点集的最小包围圆，随即弹出对

话框，显示运算时间。也可通过单击工具栏选项实现。

4、点的选择和拖动

平面点集生成好以后，或最小包围圆计算完成后，单击“操作->选择”，鼠标随即变成十字状，将鼠标移至窗口内某点，单击左键长按左键并移动鼠标，选中点随即被移动，若将圆内点移至圆外，最小包围圆随即被重新计算。

5、平移，放大，缩小

单击菜单选项“操作->平移/放大/缩小”，即可实现窗口的平移、放大、缩小功能。也可通过单击工具栏选项实现。

6、构造演示

平面点集生成好以后，“测试->最小包围圆构造演示”，随即弹出对话框，可以设置最小包围圆动态更新的时延，单击“确定”，即可实现最小包围圆的动态生成过程。该过程可以很直观的观察算法实现的过程。但对于点数较多的情况，不建议使用构造演示功能。

七、参考文献

1.《计算几何——算法与应用》（第二版）.M.de Beng, M.van Kreveld, M. Overmars, O.Schwarzkopf 著. 邓俊辉 译. 清华大学出版社. 2005.9；

2.《计算几何——算法设计与分析》（第二版）. 周培德 著. 清华大学出版社. 2005.4；

3. Solution methodologies for the smallest enclosing circle problem . Sheng Xu, Robert M. Freund, Jie Sun；

4. Dynamic planar convex hull operations in near-logarithmic amortized time. Timothy M. Chan. 1999.7.