全国第三届研究生数学建模竞赛

题 目 Ad Hoc 网络中的区域划分和资源分配问题

摘 要：

本文解决的是 Ad Hoc 网络中的区域划分和资源分配问题。

首先，针对题目的相邻两个圆公共面积比进行分析，参考喷灌模型中喷头安置模型，讨论了不同面积比下以圆心的简化分布，当公共面积比不小于 5%和 18%时圆心分别按正三角形和正四边形分布，并证明了其合理性。逆向思维，用边长为 1000×1000 的正方形覆盖圆，移动正方形使正方形区域内圆最少，以正方形与圆的面积之比最大（如果一个圆只有部分在正方形区域中，也按一个计算）为目标函数。充分考虑到图形的对称性，求得正方形内可最少填充 45 个圆，并求解到取得 45 个圆时的定义域，最后深入讨论了定义域对目标函数的影响，使求解过程具有一般性。

然后，在问题二中假设没有椭圆，以圆半径和圆心进行全局搜索并绘出关系图，求到最优预覆盖半径为 100，最优预覆盖半径和为 4500。在最优预覆盖的基础上，证明了当且仅当区域边界处（正方形边界和椭圆边界处）存在劣弧时理想覆盖半径和才可能减少，最后给出了两种求解方案。通过建立半径之和与半径和正方形平移量之间约束关系，求得最小半径和为 4327.1。

最后，采用了链路分群算法中的最大连接度分群算法和聚敛中心相结合的算法，通过引入基于距离的学习策略，找到半径之和最小的一跳覆盖区划分，收敛速度。

参赛队号 10384002

参赛密码 ________

（由组委会填写）

参赛队员姓名 蔡清波，陈云，屈小波

一、等半径区域划分模型

1.问题复述

在一个 1000×1000(面积单位)的区域内构建一个 Ad Hoc 网络，将此正方形区域用若干个半径都是 100 的圆完全覆盖，要求相邻两个圆的公共面积不小于一个圆面积的 5%，最少需要多少个圆（如果一个圆只有部分在正方形区域中，也按一个计算）。若给每个圆分配一个信道，使得有公共部分的圆拥有不同的信道，最少需要几个信道，并如何分配。若 5%改为 18%，最少需要多少个圆。

对以上两种划分，若每个公共部分中心和相应圆心各恰有一个节点，从节点集合中随机地抽掉 2%、5%、10%、15%等数量的节点后网络是否仍然连通，讨论网络的抗毁性。

2.基本假设

（1） 相邻两个圆的公共面积指两个完整的圆相交的面积，即使该圆的大部分面积位于正方形区域外也按完整的一个圆计算。

（2） 在考虑连通性和抗毁性时，只要圆用于覆盖区域都就当作一个圆处理，无论该圆是全部在覆盖区域内还是区域外。

3.符号说明

（1） D：区域的抽象表示

（2） o：点的抽象表示

（3） I：正方形区域

（4） d_0：区域 I 的边长

（5） S_D:区域 D 的面积

（6） S_I：方形 I 的面积

（7） $[x]$：小于 x 的最大整数

（8） $o_i(x_i, y_i)$：第 i 个圆的圆心

（9） R：圆的半径，$R=100$

（10）$O(o_i)$：以o_i为圆心，半径为R的圆

（11）R：圆的半径，$R=100$

（12）$S_{D(O(o_i),O(o_j))}$：第i个圆与第j个圆相交面积

（13）α：交叉面积比，表示相邻两个圆的公共面积占一个圆面积的比例

（14）$G(o_i)$：0，1 变量，以o_i为圆心的圆位于方形Ⅰ内，$G(o_i)=1$，若不位于Ⅰ内，$G(o_i)=0$

4.问题背景及分析

Ad Hoc 网络是当前网络和通信技术研究的热点之一，对于诸如军队和在野外作业的大型公司和集团来说，Ad Hoc 网络有着无需基站、无需特定交换和路由节点、随机组建、灵活接入、移动方便等特点，因而具有极大的吸引力。

本题是实际中的 Ad Hoc 网络的特定和简化情形，问题一对需要组网所涉及的一块区域，进行区域分配，用等半径的圆覆盖正方形区域，以保证区域内的圆个数最少。

第一问中由于圆的半径固定，都是100m，而且要求完全覆盖住1000×1000m的正方形区域且相邻两个圆的公共面积不小于一个圆面积的5%，(如果一个圆只有一部分在正方形区域中，也按一个计算)，现在要求的是最少需要多少个圆。题目要求圆能够完全覆盖正方形区域，所以，在覆盖后不应出现盲区（没有圆覆盖的区域）。由于正方形区域是并无特别要求，可认为是均匀分布。如果将圆的中心看作一点，因为圆半径固定，问题就归结为在保证相邻圆之间有5%区域相交和圆完全覆盖正方形区域的条件下，在正方形区域内安置圆心，使圆最少。

问题一与喷灌问题中,在均匀地面安置喷头,保证洒水均匀和喷头最少相近。根据文献《喷灌模型》，平面上喷头构成正多边形最为合理，正多边形只有正三角形、正方形和正六边形。下面就相邻圆的交叉方案进行论证。

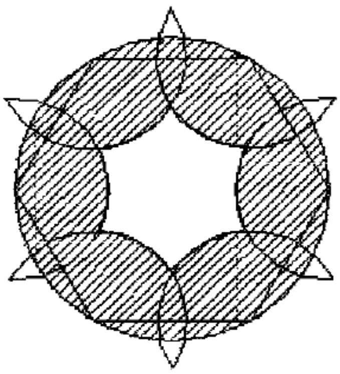


图1.4.1(a)

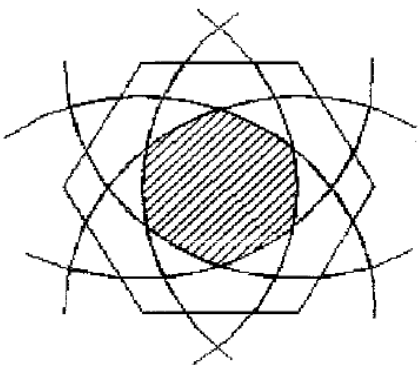


图1.4.1(b)

图 1.4.1 是按照正六边形安排圆心的示意图,显然,正六边形可以首先排除。根据文献《喷灌模型》，如果圆心间距过大，会出现如图 1.4.1(a)所示的盲区。

如果圆心间距变小，就需要需要较多的圆，所以，它显然不是最佳方案。本模型中，只需研究和比较正三角形和正四边形即可，即只需证明当采用正三角形安置圆心时，相邻圆的交叉覆盖面积不小于 5%或 18%，采用正四边形安置圆心时，相邻圆的交叉覆盖面积不小于 5%或 18%即可。

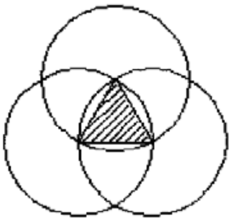


图1.4.2(a)

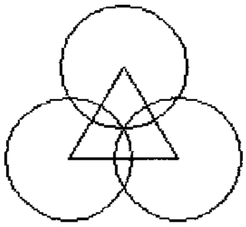


图1.4.2(b)

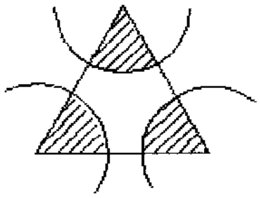


图1.4.2(c)

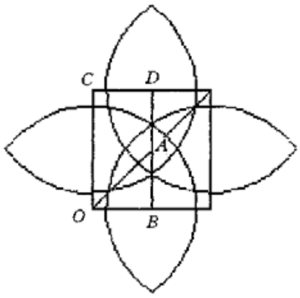


图1.4.2(d)

图1.4.2是按照正三角形和正四边形安置圆心的示意图。显然，这两种安置方式的合理性取决于交叉面积比，即 $S_{ij} \geq \alpha \pi R^2$。

设相交扇形的圆心角为θ，如图1.4.3所示

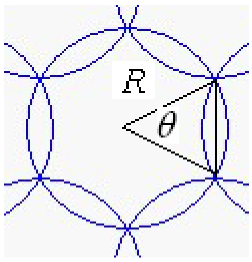


图1.4.3

$$S_{ij} = (\pi R^2 \frac{\theta}{2\pi} - \frac{1}{2} R^2 \sin\theta) \times 2$$

由 $S_{ij} \geq \alpha \pi R^2$，解得：

$$\theta - \sin\theta \geq \pi\alpha$$

由MATLAB求解得：当 $\alpha = 5\%$ 时， $\theta = 0.9968 = 57.1146^\circ$

当 $\alpha = 18\%$ 时， $\theta = 1.5655 = 89.6950^\circ$

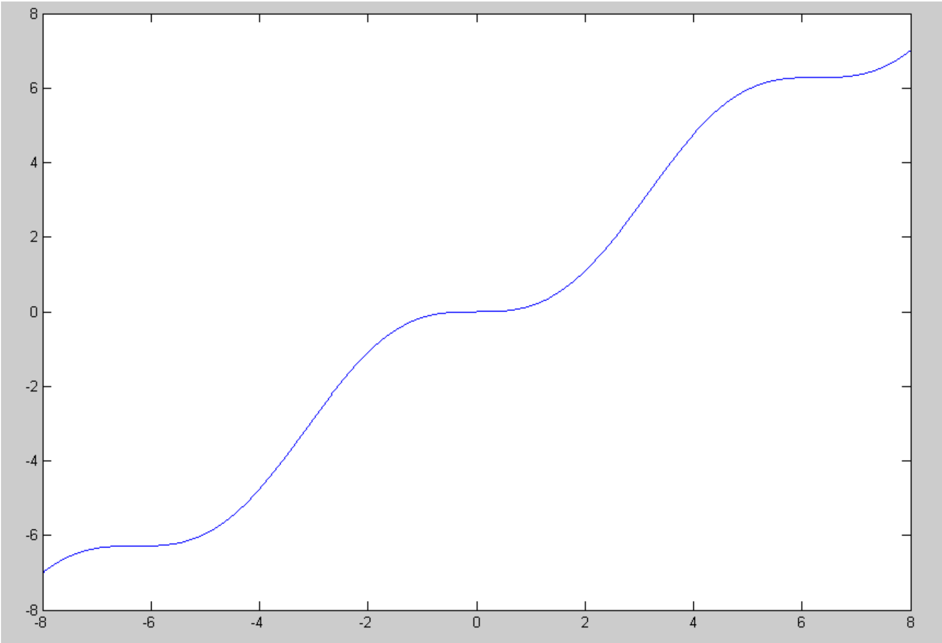


图1.4.4

图1.4.4是 $f(\theta)=\theta-\sin\theta$ 在 $[-8,8]$ 上的图形，显然，当 $\theta=60^{\circ}$ 时，$\alpha=0.0577>5\%$，但小于 18%，所以，当 $\alpha=5\%$ 时，圆心按照正三角形安置是合理的。同理，当 $\theta=90^{\circ}$ 时，$\alpha=0.1817>18\%$，所以，当 $\alpha=18\%$ 时，圆心按照正四边形安置是合理的。

从上面的证明可见，为了使得公共面积不小于一个圆面积的 5%，而且不至于出现盲区（没有圆覆盖的区域），应该让任意 3 个圆彼此互相覆盖，圆心距为

$$d=2R\cos\frac{\theta}{2}=2\times100\times\cos\frac{60^{\circ}}{2}=100\sqrt{3}$$

这时，3 个圆的公共部分聚于一点，需要的圆是最少的。

同理，如果要使相临两个圆的交叉面积不小于一个圆面积的 18%，应该让任意 4 个圆互相覆盖，圆心距为

$$d=2R\cos\frac{\theta}{2}=2\times100\times\cos\frac{90^{\circ}}{2}=100\sqrt{2}$$

这时，4 个圆的公共部分聚于一点，需要的圆是最少的。

至此，交叉面积比 $\alpha\geq5\%$ 条件已经充分满足，问题转化为在圆心按正三角形分布的情况下，如何安置圆心使圆的数目最少；交叉面积比 $\alpha\geq18\%$ 时，圆心按正四边形分布的情况下，如何安置圆心使圆的数目最少。

5. 模型建立

在确定了相邻圆交叉方案后，在图？的基础上，我们假想图？无限大，现在只需将方形区域 I 在图？上移动，使得区域 I 内的圆的数目最少。

以o_i为圆心的圆位于方形I内，$G(o_i)=1$，若不位于I内，$G(o_i)=0$

$$G(o_i)=\begin{cases}1 & O(o_i)\bigcap O(o_j)=\varnothing, O(o_i)\in I, O(o_j)\in I \\ 1 & S_{D(O(o_i),O(o_j))}\geq \alpha\pi R^2, O(o_i)\bigcap O(o_j)\neq\varnothing, O(o_i)\in I, O(o_j)\in I \\ 0 & S_{D(O(o_i),O(o_j))}< \alpha\pi R^2, O(o_i)\bigcap O(o_j)\neq\varnothing, O(o_i)\in I, O(o_j)\in I \\ 0 & O(o_i)\notin I\end{cases}$$

因为方形区域I是边长为d_0的正方形，所以，使I内圆的数目最少，即方形I的面积与区域内的圆面积之和的比最大（如果一个圆只有一部分在正方形区域中，也按一个计算）。

所以，问题一的模型为：

$$\max \quad \frac{S_{\mathrm{I}}}{\sum_i S_{O(o_i)}}$$

其中，$S_{O(o_i)}=G_i(o_i)\pi R^2$

6. 模型求解

6.1 最少需要的圆的个数

在问题一的分析中，我们已经确定了相邻圆交叉方案，在图1.4.2的基础上，我们假想图1.6.1无限大，显然，图1.6.1具有对称性。

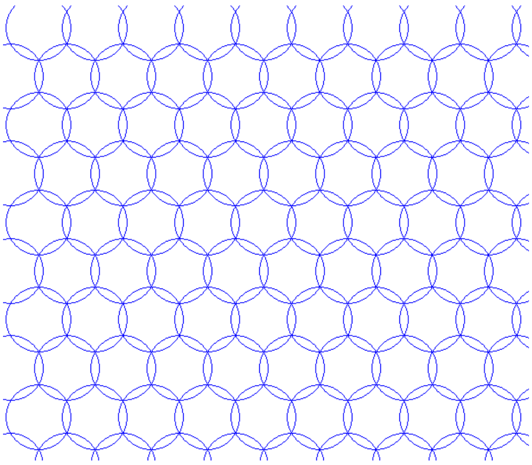


图1.6.1

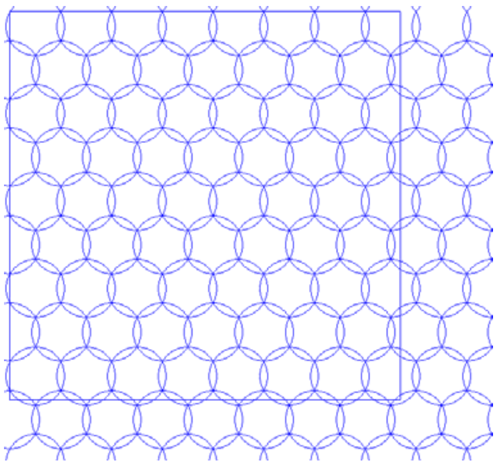


图1.6.2

为达到用最少圆覆盖1000×1000 m的方形区域I，用1000×1000 m的方形去覆盖图1.6.1中无限多个相交圆，如图1.6.2所示。现在，我们只主要将方形I沿X轴平移$\Delta x(0\leq\Delta x<\frac{R}{2})$，沿Y轴平移$\Delta y(0\leq\Delta y<\frac{R}{2})$，找到能覆盖的最少圆的位置即可，如图1.6.2所示。

（1） $\alpha=5\%$的最少需要的圆的个数

当$\alpha=5\%$时，圆心距$d=100\sqrt{3}$，在X轴上移动时，肯定能够完全分布在正方形区域I内的圆的个数N_0为：

$$N_0=\left[\frac{d_0}{d}\right]$$

因为图形是对称的，所以方形I往左移与往右移作用等效，能够移动的最大位移：

$$\Delta x_{\max}=d_0-N_0 d-kR$$

其中，$k=\left[\frac{d_0-N_0 d}{R}\right]$

所以，方形沿X轴平移$\Delta x\in\left[0,\Delta x_{\max}\right]$

代入$d=100\sqrt{3}$，$R=100$，$d_0=1000$，解得：

$$N_0=5$$

$$\Delta x_{\max}=33.9746$$

即 $$\Delta x\in\left[0,33.9746\right]$$

因为一个圆只有部分在正方形区域内时按一个圆计算，显然，如果一个圆只有一部分进入正方形区域，则$S_{O(o_i)}$将增大，不利于我们求解目标函数$\dfrac{S_{\mathrm{I}}}{\sum_i S_{O(o_i)}}$的最小值。

因为图形具有对称性，所以，方形I沿X轴平移的直线$l(x)$将位于$l_1(x_1)$和$l_2(x_2)$之间，即$x\in\left[x_1,x_2\right]$，如图1.6.3中$l(x)$所表示。

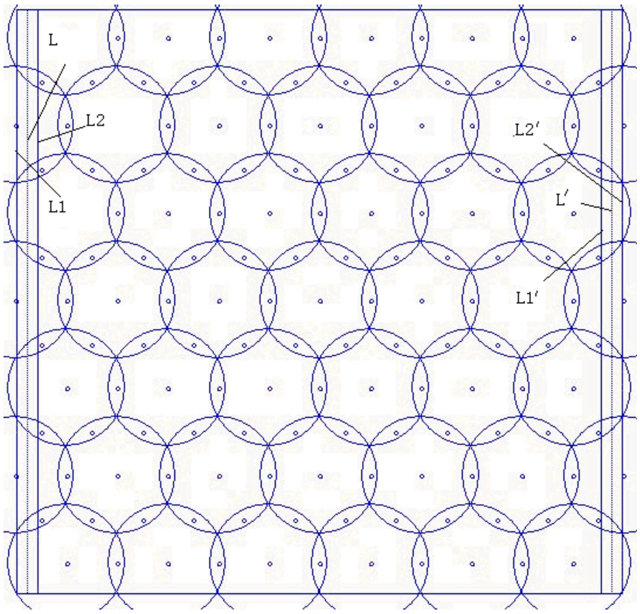


图1.6.3

图1.6.3中l_1与l_1'，l_2与l_2'，l与l'构成方形区域I。

在$x=x_1$处两圆相交，在$x=x_2+d_0$处，即l_2'处两圆相交。

显然，l往l_1左边移动，l_1左边的圆将进入方形区域I，I将覆盖更多的圆，$S_{O(o_i)}$增大，目标函数的值变小，不可能出现最优解。

l往l_2右边移动，l_2'右边的圆将进入方形区域I，I将覆盖更多的圆，$S_{O(o_i)}$增大，目标函数的值变小，不可能出现最优解。

在l_1、l_2之间移动时，I将覆盖的圆不变，所以，当$x\in[x_1,x_2]$之间时，I覆盖的圆最少，$S_{O(o_i)}$最小，目标函数的值最大，取得最优解，$[x_1,x_2]$即为最优解的定义域。

在保证水平移动在最优定义域内，方形I沿Y轴平移的直线$m(y)$将位于$m_1(y_1)$和$m_2(y_2)$之间，即$y\in[y_1,y_2]$，如图1.6.4中$m(y)$所示。

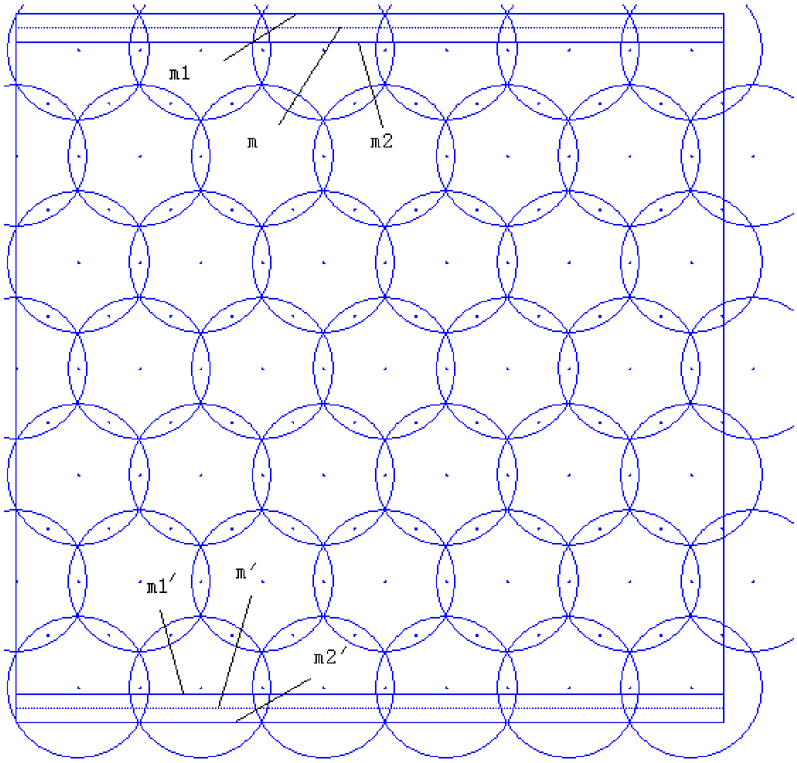


图1.6.4

在 $y=y_1$ 处两圆相交，在 $y=y_2+d_0$ 处，即 $m_2{}'$ 处两圆相交，同理可算得 $\Delta y\in[0,33.9746]$。根据 I 在X轴的移动规律，当 $y\in[y_1,y_2]$ 之间时，I 覆盖的圆最少，$S_{O(o_i)}$ 最小，目标函数的值最大，取得最优解，$[y_1,y_2]$ 即为最优解的定义域。

综上所述，区域 $D(x,y),x\in[x_1,x_2],y\in[y_1,y_2]$ 为 $\alpha=5\%$ 下的最优解区域，$\dfrac{S_{\mathrm{I}}}{\sum\limits_i S_{O(o_i)}}$ 得到最大值：

$$\left.\frac{S_{\mathrm{I}}}{\sum\limits_i S_{O(o_i)}}\right|_{\max}=141.3717$$

方形 I 覆盖的圆的最小个数为：

$$\left.\sum_i G(o_i)\right|_{\min}=45$$

事实上，问题一只需要找到方形 I 内圆的覆盖情况，在方形 I 上下移动过程中，为保证移动过程中方形I不覆盖新的圆，说得图形完全相同（如图1.6.5所示），这是由数据的特殊性造成的。

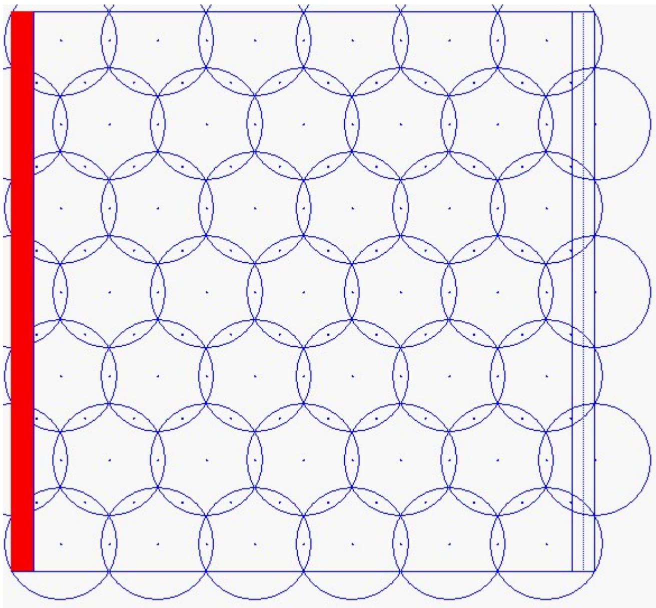


图1.6.5

覆盖情况如图1.6.6所示，方形区域表示了45个圆时的最优覆盖情况。

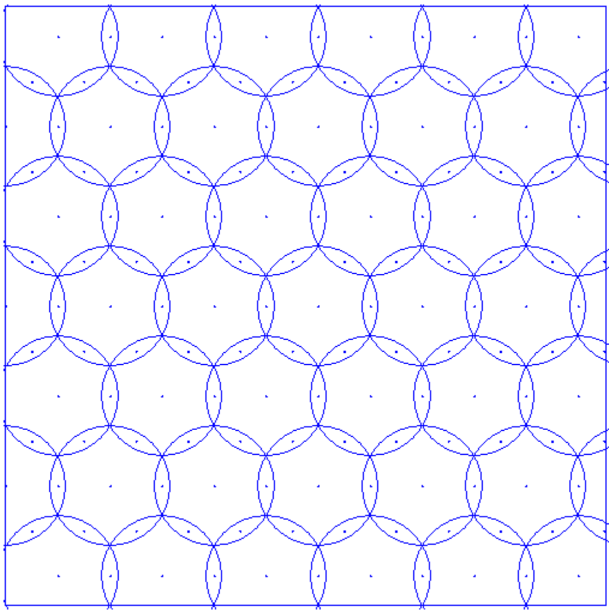


图1.6.6

（2） $\alpha=18\%$ 最少需要的圆的个数

当 $\alpha=18\%$ 时，圆心距 $d=100\sqrt{2}$ ，在X轴上移动时，肯定能够完全分布在正方形区域 I 内的圆的个数 N_0 为：

$$N_0=\left[\frac{d_0}{d}\right]=\left[\frac{1000}{100\sqrt{2}}\right]=7$$

方形 I 在X轴上能够移动的最大位移：

$$\Delta x_{\max}=d_0-N_0d-kR=1000-7\times100\sqrt{2}-0\times100=10.0505$$

即 $\Delta x \in [0,10.0505]$

同理 $\Delta y \in [0,10.0505]$

在 $x = x_1$ 处两圆相交，在 $x = x_2 + d_0$ 处，即 l_2' 处两圆相交。在 l_1、l_2 之间移动时，I 将覆盖的圆不变，所以，当 $x \in [x_1, x_2]$ 之间时，I 覆盖的圆最少，$S_{O(o_i)}$ 最小，目标函数的值最大，取得最优解，$[x_1, x_2]$ 即为最优解的定义域。在 $y = y_1$ 处两圆相交，在 $y = y_2 + d_0$ 处，即 m_2' 处两圆相交。根据 I 在X轴的移动规律，当 $y \in [y_1, y_2]$ 之间时，I 覆盖的圆最少，$S_{O(o_i)}$ 最小，目标函数的值最大，取得最优解，$[y_1, y_2]$ 即为最优解的定义域。

区域 $D(x, y), x \in [x_1, x_2], y \in [y_1, y_2]$ 为 $\alpha = 18\%$ 下的最优解区域，$\dfrac{S_{\mathrm{I}}}{\sum_i S_{O(o_i)}}$ 得到最大值：

$$\left.\frac{S_{\mathrm{I}}}{\sum_i S_{O(o_i)}}\right|_{\max} = 201.0619$$

方形 I 覆盖的圆的最小个数为：

$$\left.\sum_i G(o_i)\right|_{\min} = 64$$

覆盖情况如图1.6.7所示，方形区域表示了64个圆时的最优覆盖情况。

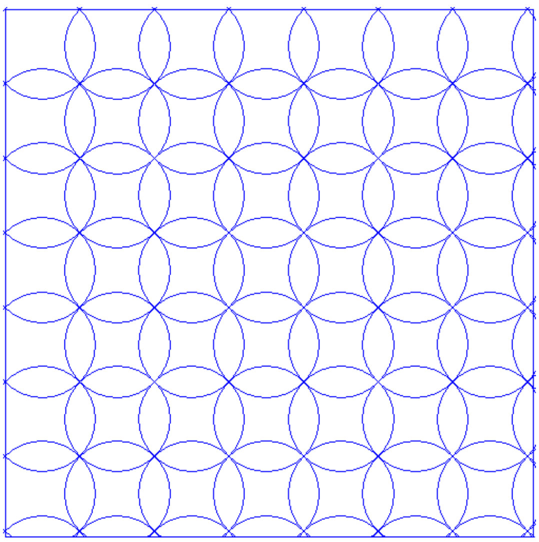


图1.6.7

6.2 网络连通性

因为节点均分布在圆内，所以，当$\alpha=5\%$时，圆心按正三角形分布，节点的总数为149个，其中圆内有4个节点的有10个圆，5个节点的有8个圆，当抽取2%时，即随机抽取3个节点，则该网络连通的概率为：

$$\frac{10}{45}\times\frac{3}{4}+\frac{35}{45}=94.44\%$$

当$\alpha=18\%$时，圆心按正四边形分布，节点的总数为176个，其中圆内有3个节点的有4个圆，4个节点的有24个圆，当抽取2%时，即随机抽取4个节点，则该网络连通的概率为：

$$\frac{4}{64}+\frac{24}{64}+\frac{36}{64}\times\frac{4}{5}=88.75\%$$

二、带椭圆的区域划分模型

1.问题复述

正方形区域中有一中心在（550，550）、长轴与正方形水平的一条边成 30 度角、长度为 410、短轴为 210 的椭圆形湖泊。节点仅能设置在地面上，假设一跳覆盖区圆的半径可以在 75~100 间随意选择，两个面积不等的圆相交，它们之间的公共面积应不小于大圆面积的 5%，其他假设同（1），研究使全部圆半径之和为最小的区域分划和信道分配方案。

2.基本假设

（1） 相邻两个圆的公共面积指两个完整的圆相交的面积，即使该圆的大部分面积位于正方形区域外也按完整的一个圆计算。

（2） 如果一个圆只有部分位于目标区域内，当对半径求和时，按照一个半径计算。

3.符号说明

（1） D：区域的抽象表示

（2） S_D:区域D的面积

（3）$T(D)$：区域D内所有圆的半径之和（如果一个圆只有一部分在正方形区域D中，也按一个计算）

（4） $T_{\min}(D)$：区域D内所有圆的半径之和的最小值

（5） I：正方形区域

（6） d_0：区域I的边长

（7） E：椭圆区域

（8） $\overline{E}$：正方形区域I内，椭圆区域E外的区域，即$\overline{E}\cap E=\varnothing,\overline{E}\cup E=I$

（9） $r_i(x_i,y_i)$：第i个圆的半径，$75\le r_i\le 100$

（10）$o_i(x_i,y_i)$：第i个圆的圆心

（11）$O(o_i,r_i)$：以o_i为圆心，半径为r_i的圆

（12）R：表示预覆盖的圆半径

（13）$\widetilde{R}$：表示最优预覆盖时的半径

（14）N：表示预覆盖的圆的个数

（15）$\widetilde{N}$：表示最优预覆盖时圆的个数

（16）$S_{D(O(o_i),O(o_j))}$：第i个圆与第j个圆相交面积

（17）$\widetilde{O}(\tilde{o}_i,\widetilde{R})$：表示最优预覆盖时圆心为$\tilde{o}_i$，半径长度为$\widetilde{R}$的圆

（18）$U\left(o_i,r_i,D\right)$：圆$O(o_i,r_i)$在区域D内的弧长

（19）$J\left(o_i,r_i,D_1,D_2\right)$：劣弧判断函数

$$J\left(o_i,r_i,D_1,D_2\right)=\begin{cases}1 & U\left(o_i,r_i,D_1\right)\le U\left(o_i,r_i,D_2\right)\\ 0 & U\left(o_i,r_i,D_1\right)>U\left(o_i,r_i,D_2\right)\end{cases}$$

当圆$O(o_i,r_i)$在区域D_1内的弧长小于等于它在区域D_2内的弧长时，$J=1$

当圆$O(o_i,r_i)$在区域D_1内的弧长大于其在区域D_2内的弧长时，$J=0$

4.问题分析

正方形区域I有一中心在（550，550）、长轴与正方形水平的一条边成30度角、长度为410、短轴为210的椭圆形湖泊(如图2.4.1所示),节点仅能设置在地面上。问题二的目标是使全部圆半径之和为最小，显然，半径之和受到三个因素的影响：I内圆的个数，各个圆的半径，以及圆心所在位置。

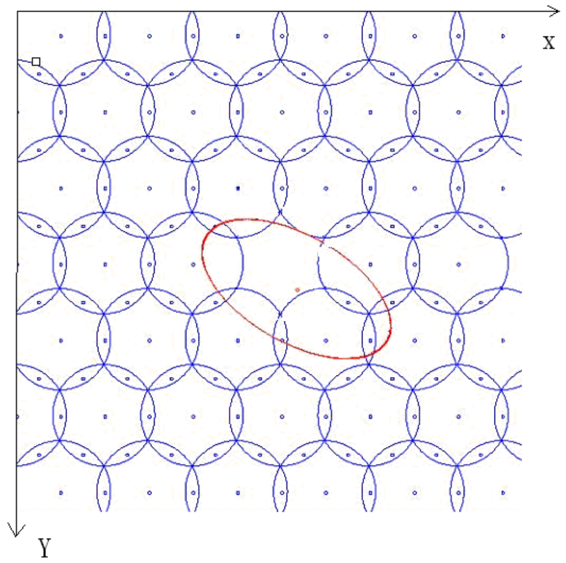


图 2.4.1

（1）区域内没有椭圆的分析（预覆盖）

我们首先分析圆半径对半径之和的影响，现假定区域内没有椭圆，我们分别用 $R_i = 75$ 和 $R_i = 100$ 的圆进行覆盖。根据问题一的分析可知，在 $\alpha = 5\%$ 时圆心成正三角形分布是最合理的。当 $R_i = 75$ 时，按照问题一的解法，可求得正方形区域 I 内的最少的圆个数为：

$$\sum_i G(o_i)\Big|_{\min} = 85$$

所以，当 $R_i = 75$ 时，$T_{\min} = \sum_i R_i G(o_i)\Big|_{\min} = 85 \times 75 = 6375$

当 $R_i = 100$ 时，$T_{\min} = \sum_i R_i G(o_i)\Big|_{\min} = 45 \times 100 = 4500$

根据问题一的求解，我们假定半径 $R_i \in [75,100]$ 并且可调，通过确定肯定能够完全分布在正方形区域 I 内的圆的个数 N_0 和可行解区域 $\Delta x \in [0, \Delta x_{\max}]$ 和 $\Delta y \in [0, \Delta y_{\max}]$ 进行全搜索，半径变化步长 $\Delta r = 0.1$，$R_i = 75 + i \times 0.1 \times k \quad (0 \le k \le 250, k \in Z)$

图 2.4.2 是表示的是半径与区域 I 内的圆的个数的关系（数据），图 2.4.3 示的是半径与区域 I 内的圆的半径之和的关系。可以看出，当 $k = 250, R_i = 100$ 时，半径之和 T 取得最小值 4500。

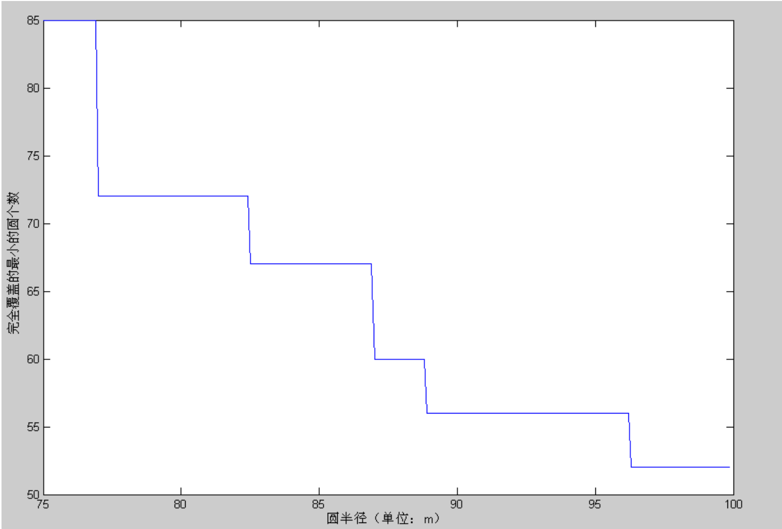


图 2.4.2

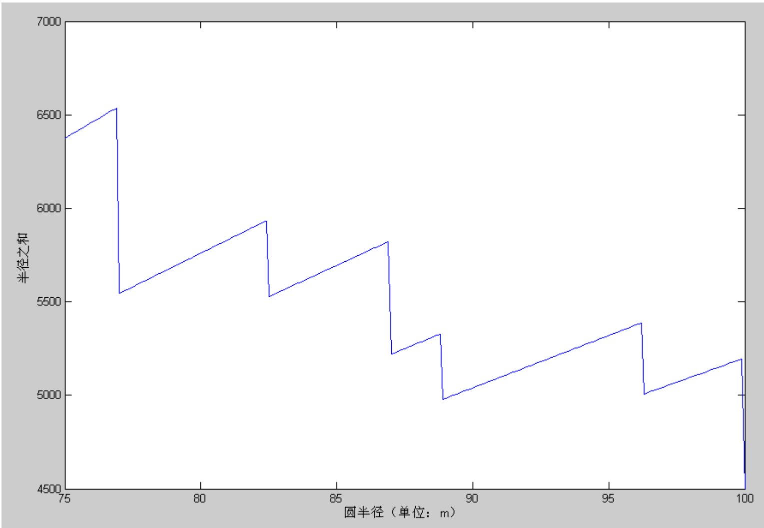


图 2.4.3

从图 2.4.2 可以看出，在某些不等的半径下，覆盖区域I需要的个数相同，当$R_i = 90, R_i = 95$时$\sum_i G(o_i)\Big|_{\min} = 56$，最少圆数相等。可见，用圆填充到区域$I$内时，如果$\sum_i G(o_i)\Big|_{\min}$相等，应尽量小的半径，以使半径之和$T$最小。

除了考虑覆盖区域I的圆的半径之外，椭圆的存在以及边界上覆盖的不完全圆也会对圆半径之和造成影响，下面分别讨论二者的影响。

（2）区域内有椭圆的分析（椭圆覆盖）

当区域I内有椭圆时，上述分析中的圆只可能与椭圆有三种关系，圆完全在椭圆内（图2.4.4中圆A），圆完全不在椭圆内（图2.4.4中圆D），圆与椭圆相交（图2.4.4中圆B和圆C）。

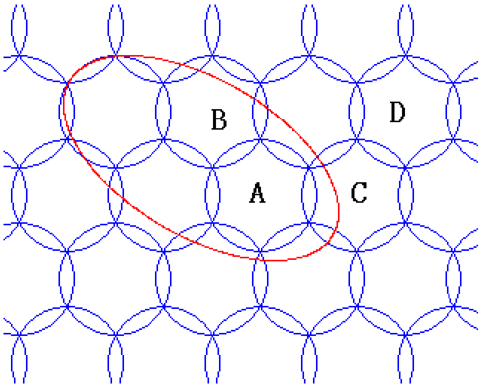


图2.4.4

显然，当圆完全在椭圆内时，只需从椭圆内去除掉该圆即可。当圆完全在椭圆外时，根据前面的分析，圆的半径应越大越好。

椭圆与圆相交只可能有三种情况：圆在区域$\overline{E}$内的弧长$\overset{\frown}{MPN}$大于在椭圆E内的弧长$\overset{\frown}{MP'N}$，即 $0<\dfrac{\overset{\frown}{MPN}}{\overset{\frown}{MP'N}}\le 1$，此为劣弧（如图 2.4.5(a)中圆 B 所示）；圆在区域$\overline{E}$内的弧长$\overset{\frown}{MPN}$大于等于在椭圆E内的弧长$\overset{\frown}{MP'N}$，即 $\dfrac{\overset{\frown}{MPN}}{\overset{\frown}{MP'N}}>1$，此为优弧（如图 2.4.5 (b)圆 C 所示）。

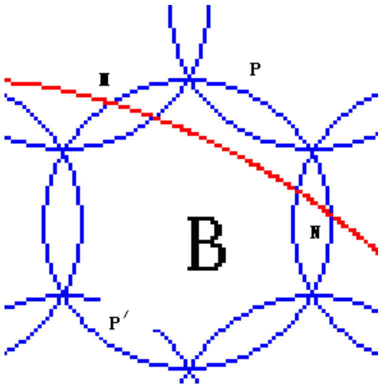


图 2.4.5(a)

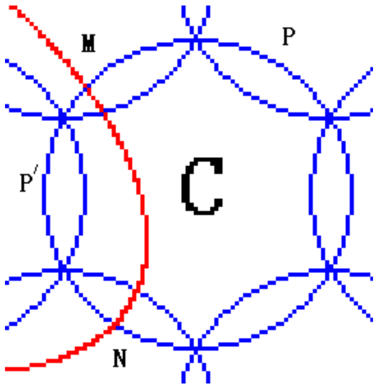


图 2.4.5 (b)

对于图？中的劣弧相交情况，因为填充区域$\overline{E}$的圆半径较大，所以，在保证交叉面积比和无盲区覆盖的情况下，可采用尽量小半径的圆代替原来的圆，以使半径之和最小，如图 2.4.6 中圆 B′ 所示。

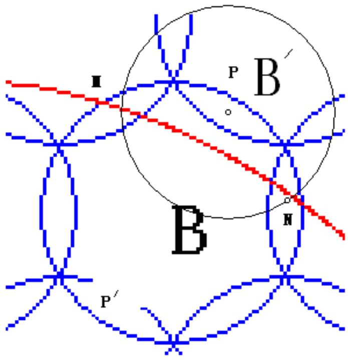


图 2.4.6

对于图 2.4.6 中的优弧相交情况，显然不能用半径小的圆进行代替。假设用半径更小的圆来代替，显然不能保证交叉面积比和无盲区覆盖。所以，对于与椭圆相交的圆，如果是优弧相交，仍然以（1）中的圆的半径作为该圆半径。

下面讨论替换圆的圆心和半径的确定。

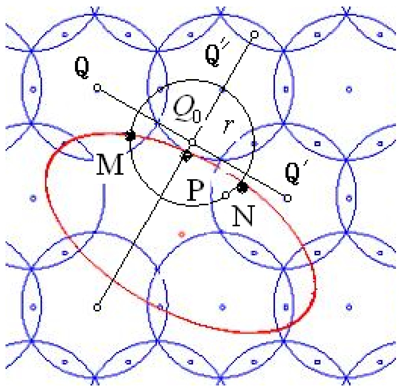


图 2.4.7

如图 2.4.7 所示，圆Q、Q'和Q''是与劣弧$\widehat{MNP}$相关的三个圆，为保证无盲区覆盖，显然将圆心Q_0放置在过Q''做QQ'的中垂线，中垂线交于QQ'的地方即为Q_0说在处，也就是QQ'的中点处。现调整圆的半径r，以使圆Q_0与圆Q、Q'和Q''交叉面积比α均大于等于5%。为保证圆半径之和最小，取$\alpha = 5\%$即可。

（3）边界上的圆对半径和的影响（边界覆盖）

从（2）中关于圆弧的分析可以看出，如果一个圆仅有一段劣弧在区域I内，该圆是需要调整半径和圆心的。从图？可以看出，这样的圆在边界上同样存在，根据 2）中的分析，圆心Q_0放置在QQ'的中点处，整圆的半径r，以使圆Q_0与圆Q、Q'和Q''交叉面积比α均大于等于5%（如图 2.4.8 所示）。为保证圆半径之和最小，取$\alpha = 5\%$即可。

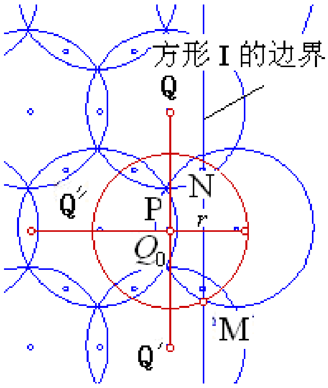


图 2.4.8

（4）平移对最小半径和的影响（半径调整）

前面的讨论都是在问题一的求解基础上进行讨论的，事实上，从问题一答案中圆覆盖正方形区域I的情况来看，问题一存在最优解的定义域，并不是唯一的。从最优解的定义域可以看出，图形？中正方形区域I可以继续向左移动，如图？中红色虚线所示，区域的左右边界优弧$\widehat{MPN}$可以变成劣弧$\widehat{M'PN'}$，根据上述（2）和（3）的分析，可以用半径r更小的圆进行覆盖劣弧$\widehat{M'PN'}$所圆的区域，以使半径和T最小。

5. 模型的建立

通过分析可以看出，我们首先使正方形区域I（不带椭圆）内用等大的半径圆进行覆盖，并使半径之和最小，得到最优预覆盖半径$\widetilde{R}$和最优预覆盖圆的个数$\widetilde{N}$。然后，去掉完全在椭圆E内部的圆，在圆半径r可变的情况下，保证$\overline{E}$中圆的个数和$\sum_i G(o_i)=\widetilde{N}$不变，在问题一的最优解定义域内水平移动正方形$I$，使得$\overline{E}$的边界处出现更多的劣弧，并尽量用半径小的圆去覆盖$\overline{E}$。

问题二的模型：

$$\min \quad T(\overline{E})=\widetilde{N}\widetilde{R}+\sum_i J\left(o_i(x_i,y_i),r_i,E,\overline{E}\right)\Delta r(o_i(x_i,y_i),\Delta x,\Delta y)$$

St.

$$\begin{cases} \widetilde{N} = \sum_i G(o_i) \\ 75 \le r_i \le 100 \\ 75 \le r_i + \Delta r \le 100 \\ 0 \le \Delta x \le \Delta x_{\max} \\ 0 \le \Delta y \le \Delta y_{\max} \end{cases}$$

其中， $\Delta r(o_i(x_i, y_i), \Delta x, \Delta y) = G(o_i(x_i + \Delta x, y_i + \Delta y)\left(r_i(x_i + \Delta x, y_i + \Delta y) - \widetilde{R}\right)$

$$\Delta x_{\max} = d_0 - N_0 d - \left[\frac{d_0 - N_0 d}{R}\right] R$$

$$\Delta y_{\max} = d_0 - N_0 d - \left[\frac{d_0 - N_0 d}{R}\right] R$$

$$J\left(o_i(x_i, y_i), r_i, E, \overline{E}\right) = \begin{cases} 1 & U\left(o_i, r_i, E\right) \le U\left(o_i, r_i, \overline{E}\right) \\ 0 & U\left(o_i, r_i, E\right) > U\left(o_i, r_i, \overline{E}\right) \end{cases}$$

$$G(o_i(x, y)) = \begin{cases} 1 & O(o_i) \cap O(o_j) = \varnothing, O(o_i) \in \overline{E}, O(o_j) \in \overline{E} \\ 1 & S_{D(O(o_i),O(o_j))} \ge \alpha \pi R^2, O(o_i) \cap O(o_j) \ne \varnothing, O(o_i) \in \overline{E}, O(o_j) \in \overline{E} \\ 0 & S_{D(O(o_i),O(o_j))} < \alpha \pi R^2, O(o_i) \cap O(o_j) \ne \varnothing, O(o_i) \in \overline{E}, O(o_j) \in \overline{E} \\ 0 & O(o_i) \notin \overline{E} \end{cases}$$

6. 模型的求解

（1）不考虑沿 X 轴和沿 Y 轴平移

利用问题 1 的方法，当 $\alpha \ge 5\%$ 时，根据半径变化步长 $\Delta r = 0.1$ 的全局搜索可以求得：

$$T_{\min}\left(I\right) = \sum_i G(o_i)\widetilde{R}\Big|_{\min} = 4500$$

$$\widetilde{N} = \frac{T_{\min}\left(I\right)}{\widetilde{R}} = 45$$

$$\widetilde{R} = 75$$

因为劣弧只可能出现在椭圆 E 的边界处和方形 I 的边界处，设椭圆方程为 $y = e\left(x\right)$，正方形区域 I 内第 i 个圆的圆方程为 $\left(x - x_i\right)^2 + \left(y - y_i\right)^2 = \widetilde{R}^2$，由

$$\begin{cases} y = e(x) \\ (x-x_i)^2 + (y-y_i)^2 = \widetilde{R}^2 \end{cases}$$

可以联立解得椭圆与第 i 个圆的交点 $M_i\left(\widetilde{x}_i, \widetilde{y}_i\right)$ 和 $N_i\left(\widetilde{x}_i', \widetilde{y}_i'\right)$，通过圆心 $o_i\left(x_i, y_i\right)$，求出 $\overset{\frown}{M_iP_iN_i}$ 和 $\overset{\frown}{M_iP_i'N_i}$，其中 $\overset{\frown}{M_iP_iN_i} \in \overline{E}, \overset{\frown}{M_iP_i'N_i} \in E$ 。由于手工计算复杂，我们采用 MATLAB 进行数值求解，得到椭圆边界的圆为劣弧的有两个，如图 2.6.1 所示中的圆 $\widetilde{O}(o_1, \widetilde{R})$ 和圆 $\widetilde{O}(o_2, \widetilde{R})$，但因圆 $\widetilde{O}(o_2, \widetilde{R})$ 的劣弧非常接近半圆弧，因此计算中不予考虑，认为是优弧。

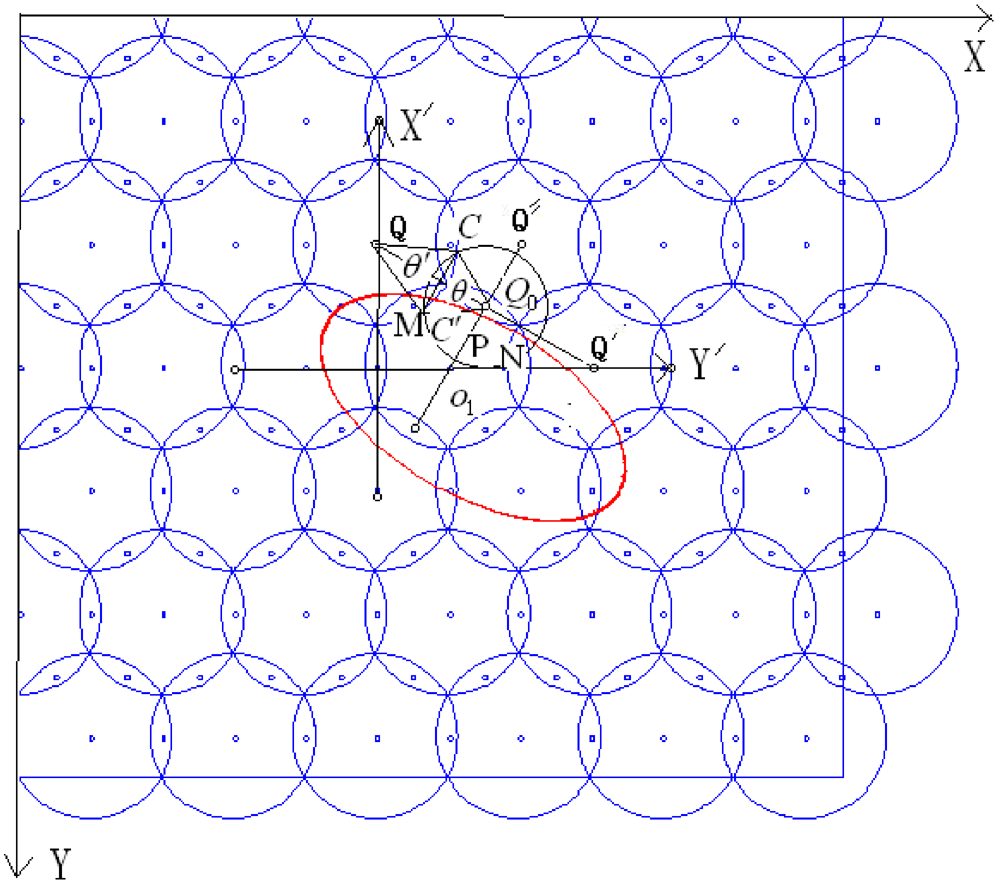


图 2.6.1

下面以处理 $\widetilde{O}(o_1, \widetilde{R})$ 的劣弧 $\overset{\frown}{M_iP_iN_i}$ 为例进行计算。

根据前面分析，替代圆 $O(Q_0\left(x_0, y_0\right), r)$ 的圆心 $Q_0\left(x_0, y_0\right)$ 在圆心 $Q\left(x_Q, y_Q\right)$ 和 $Q'\left(x_{Q'}, y_{Q'}\right)$ 的中心处，所以：

$$\begin{cases} x_0 = \dfrac{x_Q + x_{Q'}}{2} \\ y_0 = \dfrac{y_Q + y_{Q'}}{2} \end{cases}$$

用 $S_{D(O(Q),O(Q_0))}$ 表示圆 $O(Q)$ 和圆 $O(Q_0)$ 相交部分的面积，$S_{D(O(Q'),O(Q_0))}$ 表示圆 $O(Q')$ 和圆 $O(Q_0)$ 相交部分的面积，$S_{D(O(Q''),O(Q_0))}$ 表示圆 $O(Q'')$ 和圆 $O(Q_0)$ 相交部分的面积。为保证交叉面积比 α 大于等于大圆面积的5%，必须有：

$$\begin{cases} S_{D(O(Q),O(Q_0))} \geq \alpha\pi\widetilde{R}^2 = 5\% \times 100^2\pi \\ S_{D(O(Q'),O(Q_0))} \geq \alpha\pi\widetilde{R}^2 = 5\% \times 100^2\pi \\ S_{D(O(Q''),O(Q_0))} \geq \alpha\pi\widetilde{R}^2 = 5\% \times 100^2\pi \end{cases}$$

以 $S_{D(O(Q),O(Q_0))}$ 计算为例，设圆 $O(Q)$ 和圆 $O(Q_0)$ 相交于 $C(x_c, y_c)$ 和 $C'(x_{c'}, y_{c'})$ 两点，设弦 $|CC'|$ 对应圆心 Q 的圆心角为 θ'，对应圆心 Q_0 的圆心角为 θ，可得：

$$\theta' = \arccos(\frac{|CC'|^2 - 2\widetilde{R}^2}{2\widetilde{R}^2})$$

$$\theta = \arccos(\frac{|CC'|^2 - 2r^2}{2r^2})$$

$$S_{D(O(Q),O(Q_0))} = \frac{\theta'\widetilde{R}^2}{2} - \frac{1}{2}\widetilde{R}^2\sin\theta'$$

$$S_{D(O(Q),O(Q_0))} = \frac{\theta'\widetilde{R}^2}{2} - \frac{1}{2}\widetilde{R}^2\sin\theta' + \frac{\theta r^2}{2} - \frac{1}{2}r^2\sin\theta$$

即有 $$\frac{\theta'\widetilde{R}^2}{2} - \frac{1}{2}\widetilde{R}^2\sin\theta' + \frac{\theta r^2}{2} - \frac{1}{2}r^2\sin\theta \geq \alpha\pi\widetilde{R}^2$$

另外，为保证代替的小圆 $O(Q_0)$ 能完全覆盖原来圆 $O(o_1)$ 的覆盖区域，必须有

$$r \geq |MQ_0|$$

其中，$|MQ_0| = \sqrt{(x_M - x_Q)^2 + (y_M - y_Q)^2}$

所以，替代圆半径 r 的约束条件为：

$$\begin{cases} 75 \le r \le 100 \\ \dfrac{\theta' \widetilde{R}^2}{2} - \dfrac{1}{2}\widetilde{R}^2 \sin\theta' + \dfrac{\theta r^2}{2} - \dfrac{1}{2} r^2 \sin\theta \ge \alpha\pi\widetilde{R}^2 \\ r \ge \sqrt{\left(x_M - x_Q\right)^2 + \left(y_M - y_Q\right)^2} \end{cases}$$

因为替代圆的圆心Q_0位于QQ'的中点上，圆$O(Q)$和圆$O(Q')$关于Q_0Q''对称，所以此约束条件可以同时使得$S_{D(O(Q'),O(Q_0))} \ge \alpha\pi\widetilde{R}^2 = 5\% \times 100^2\pi$。同理，为保证$S_{D(O(Q''),O(Q_0))} \ge \alpha\pi\widetilde{R}^2 = 5\% \times 100^2\pi$，圆$O(Q_0)$和圆$O(Q'')$之间也应当满足这样的关系式。

由于过程中公式推倒复杂，我们用 MATLAB 求解得到，在不考虑沿 X 轴和沿 Y 轴平移时，求得：

$$T_{\min}\left(\overline{E}\right) = T(\overline{E})\Big|_{\min} = 4402.1$$

其中，区域$\overline{E}$右边界的劣弧处的替代圆有 3 个，半径均为 75.7

椭圆区域E边界的劣弧处的替代圆有 1 个，半径为 75

除此之外，区域$\overline{E}$的上下和左边界处的圆均为优弧，椭圆区域E边界处的圆均为优弧。

（2）考虑沿 X 轴和沿 Y 轴平移

根据问题二的模型和对问题二的分析，沿 X 轴和沿 Y 轴平移是为了使方形区域I的边界出现更多的劣弧，目标是使目标函数$T(\overline{E})$最小，但平移的量Δx和Δy受到$\Delta x \in \left[0, \Delta x_{\max}\right]$和$\Delta y \in \left[0, \Delta y_{\max}\right]$的约束，同时，为保证完全覆盖，移动之后的圆必须使与相邻三个圆的交叉面积大于大圆的5%，必须满足上面的约束条件，并不断调整替代圆的半径r，使目标函数$T(\overline{E})$最小。

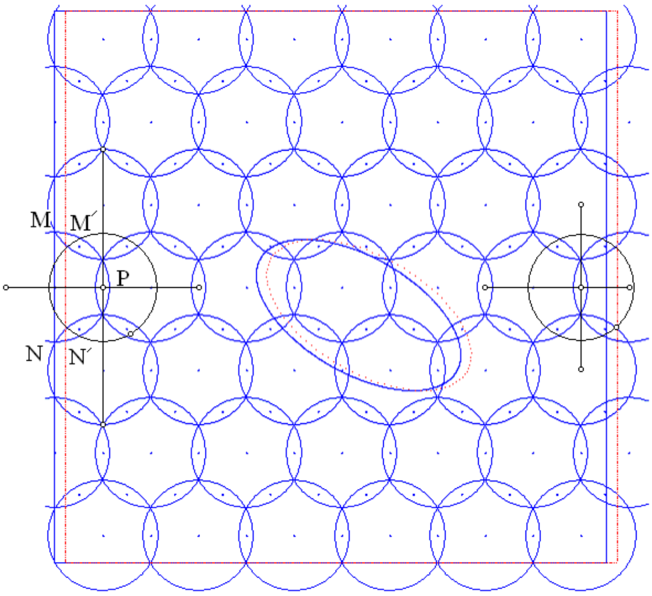


图 2.6.2

从图 2.6.2 中可以看出，方形区域I往右边移动一个小距离，原来的半圆$\overset{\frown}{MPN}$变为劣弧$\overset{\frown}{M'PN'}$，可采用比 100 小的圆进行逼近。

考虑到此题计算的复杂性，手工计算量大，但实际上变量Δx、Δy和r变化的范围并不大，可以完全使用全局搜索在短时间内求解。因此，我们对可行解区域的变化值Δx，Δy，半径r进行全局搜索。其中，Δx、Δy和r的搜索步长均设置为 0.1，可得半径变化步长$\Delta r = 0.1$， $r_i = 75 + i \times 0.1 \times k \quad (0 \le k \le 250, k \in Z)$

这样可以使我们的搜索结果精度达到：

$$\frac{0.1}{100} \times 100\% = 0.1\%$$

其中， $\Delta x \in [0, \Delta x_{\max}]$、 $\Delta y \in [0, \Delta y_{\max}]$和$r \in [75, 100]$

搜索的目标是使标函数$T(\overline{E})$最小。

事实上，根据题目图形的特征，仅仅只有图？中方形区域I的左右边界存在劣弧，通过 MATLAB 仿真，得到：

$$T_{\min}(\overline{E}) = \left. T(\overline{E}) \right|_{\min} = 4327.1$$

其中，区域$\overline{E}$右边界的劣弧处的替代圆有 3 个，半径均为 75.7

区域$\overline{E}$右边界的劣弧处的替代圆有 3 个，半径均为 75

椭圆区域E边界的劣弧处的替代圆有 1 个，半径为 75

除此之外，区域$\overline{E}$的上下边界处的圆均为优弧，椭圆区域E边界处的圆均为

优弧。

综上所述，为使得半径之和T越小，在圆半径r可变和有椭圆的情况下，在$\overline{E}$中圆的个数和$\sum_i G(o_i)$不变的时，在问题一的最优解定义域内水平移动正方形I，

使得$\overline{E}$的边界处出现更多的劣弧，尽量用半径小的圆去覆盖$\overline{E}$。

三、一跳覆盖区模型

1.问题复述

正方形区域中有一中心在（550，550）、长轴与正方形水平的一条边成 30 度角、长度为 410、短轴为 210 的椭圆形湖泊。节点仅能设置在地面上，假设一跳覆盖区圆的半径可以在 75~100 间随意选择，两个面积不等的圆相交，它们之间的公共面积应不小于大圆面积的 5%，研究使全部圆半径之和为最小的区域分划和信道分配方案。

2.基本假设

（1）在一跳覆盖区内的节点传输不需要中间节点
（2）湖泊中不能节点不存在
（3）一个节点可以同时属于不同的一跳覆盖区

3.符号说明

（1）a_{ij}：节点i与节点j的邻接度，当第 i 个节点与第 j 个节点不邻接时，$a_{ij}=0$，

否则$a_{ij}=1$

（2）A：网络的拓扑结构信息矩阵

（3）$N_{total}(i)$：包括节点 i 本身以及与其所有相邻节点的集合

（4）$V=\{1,2,...,N\}$：网络中N个节点的集合，即网络总节点集

（5）$S(\overline{E})$：区域$\overline{E}$内一跳覆盖区半径之和

4. 问题分析

假设一个比较分散的区域内有 N 个节点，其标识为 1，2，3，……，N；采用分布式控制方案。选取一个节点，通过探询方案方式获得与其临近节点的信息，在广播一个探询信号时，凡是侦听到此信息的节点都向广播探询信号的节点发送一个应答信号。这样便得到一个邻接矩阵：

$$A=\left[a_{ij}\right],\ \mathrm{a_{ij}}\in\{o,1\},\ \mathrm{i,j}=1,2,\ldots,N$$

其中，当第 i 个节点与第 j 个节点不邻接时，$a_{ij}=0$；当第 i 个节点与第 j 个节点邻接时，$a_{ji}=1$，且 $a_{ji}=1$（i，j=1，2，……，N）。

邻接矩阵 A 反映了网络的拓扑结构信息，由上式可以看出，矩阵中的元素由 0 和 1 构成，当 $a_{ji}=0$ 时表示节点 i 和节点 j 不相邻；反之，当 $a_{ji}=1$ 时，表示节点与第 j 个节点相邻。在已知网络拓扑结构信息后，就可以进行分群组网了。

设 $V=\{1,2,\ldots,N\}$ 为网络中 N 个节点的集合，即网络总节点集。$N_{adj}(i)$ 表示与节点 j 相邻节点的集合，$N_{total}(i)$ 表示包括节点 i 本身以及与其所有相邻节点的集合，即 $N_{total}(i)=N_{adj}(i)\bigcup\{i\}$。$C_i$ 表示第 i 个链路分群。

采用链路分群算法（LCA，Linked Cluster Algorithm）时，任意选取一个节点作为第 1 个群首，例如选节点 i 为第 1 个群首，则节点集 $N_{total}(i)$ 中所有的节点均处于第 1 个链路分群 C_i 中，即这些节点都成为群 C_i 的成员。之后从节点集 $N-N_{total}(i)$ 中再选取一个节点 j 作为第 2 个群首 C_j，这样节点集 $N_{total}(j)\subset 3N-N_{total}(i)$ 中的节点构成群 C_j。依次类推，当进行到每一个节点都成为某个群的成员后，链路分群划分过程结束。然后再为群与群之间选择网关节点，以便构成骨干网。

5. 模型建立

$$\min \quad S(\overline{E})=\sum_i G(o_i)r_i(o_i)$$

St.

$$\begin{cases} 75 \le r_i(o_i) \le 100 \\ G(o_i) = \begin{cases} 1 & O(o_i) \bigcap O(o_j) = \varnothing, O(o_i) \in \overline{E}, O(o_j) \in \overline{E} \\ 1 & S_{D(O(o_i),O(o_j))} \ge \alpha\pi R^2, O(o_i) \bigcap O(o_j) \ne \varnothing, O(o_i) \in \overline{E}, O(o_j) \in \overline{E} \\ 0 & S_{D(O(o_i),O(o_j))} < \alpha\pi R^2, O(o_i) \bigcap O(o_j) \ne \varnothing, O(o_i) \in \overline{E}, O(o_j) \in \overline{E} \\ 0 & O(o_i) \notin \overline{E} \end{cases} \end{cases}$$

6.模型的求解

（1）最大连接度分群算法

算法思想：最大连接度分群算法借鉴了 Internet 中选择路由器的方法，其原则是尽量减少路由器的数目，因此该算法的目标是尽量减少群的树木。节点之间通过交互控制消息知道其邻居节点的树木，该节点和其相邻节点中具有最大度的节点被选为群首，当度数相同时则选择 ID 最小的节点作为群首，群首的一跳邻居节点则成为该群的普通成员节点，反复进行以上过程，知道所有节点够加入某个群。其算法的主要策略与方法如下。

度数最大原则，即按照节点的度数最大原则选择群首节点。节点 i 的度数 $DN(i)$ 定义为：

$$DN(i) = (\sum_{j=1}^{N} a_{ij}) - 1$$

其中 $i = 1, 2, ..., N$ 。如果第 $k(k \in V)$ 个节点的度数 $DN(k)$ 满足如下式子

$$DN(k) = \max_{i \in V}[DN(i)]$$

则节点 k 被选为第 1 个群首 H_1。令 $k' \in V - N_{total}(i)$， $i' \in V - N_{total}(i)$ 如果

$$DN(k') = \max_{i' \in V - N_{total}(i')}[DN(i')]$$

则节点 k’被选为第 2 个群首 H_2，依次类推，知道链路分群划分完毕。

（2）度数最小原则，即在选取网络网关节点时，采取度数最小的优化策略，具体包括：

1）当两个相邻的链路分区交叠，且具有公共节点，同时公共节点多于一个时，按照度数最小原则选其中的一个节点作为该两个群之间的网关节点。

2）当两个链路分群相邻但不交叠时，则要从这两个群中各选取一个节点构成网关节点对，这时先选择最小的两个节点读数之和。

这种算法既能使网络的性能达到优化，又有利于实现结果设计。

（3）算法描述：每个节点广播它可以听到的所有节点的列表（包括它自己）。

1）在其所有尚未被覆盖的邻居节点中具有最大连接度的节点被推举为群首

2）尚未推举出其他群首的节点为尚未被覆盖节点，否则为已覆盖节点。

3）已推举其他节点为群首的节点不应再作为群首。

图 3.1 是按照最大连接度分群算法得出的一跳覆盖区分布情况，其中，一跳覆盖区中心连接该区内的所有节点。可以看出，由于邻接矩阵是根据定长的半径求解出来的，所以，最大连接度分群算法得到的一跳覆盖区肯定满足完全覆盖，但并不保证 5%的覆盖比例和一跳覆盖区半径之和最小。

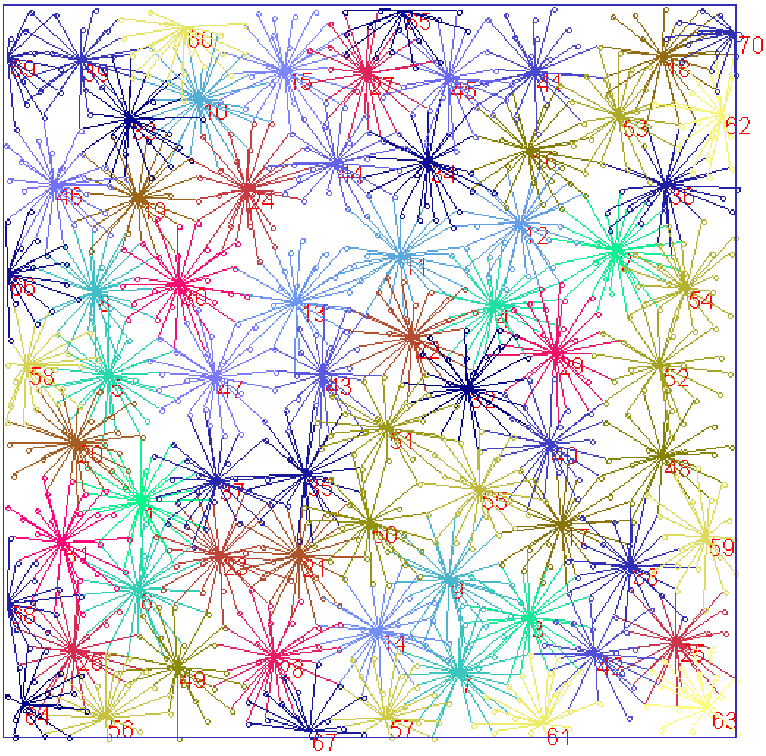
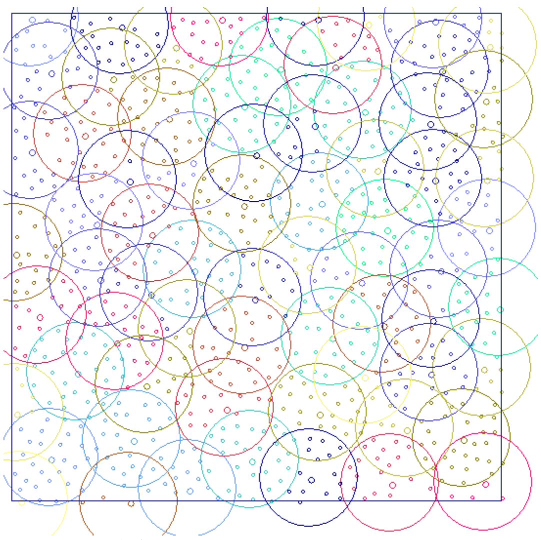


图 3.1

因此，我们对半径引入学习算法，通过聚敛分类，形成自适应分群算法，找到半径之和一跳覆盖区中心。



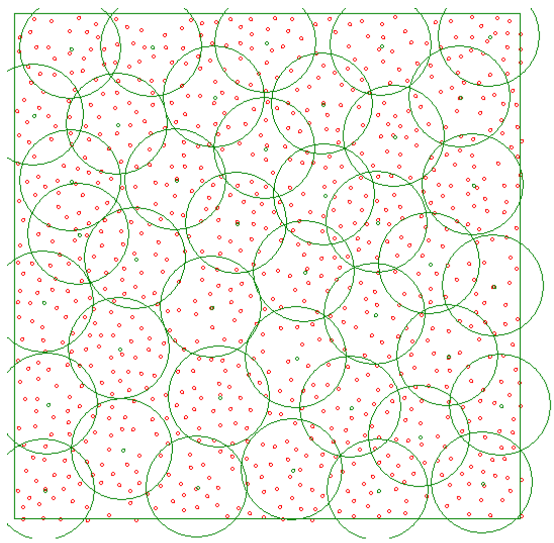


图 3.2　　　　图 3.3

图 3.2 是引入学习机制的一跳区覆盖图，图 3.3 是采用 40 个聚类中心的结果，可见，一跳区个数比最大连接度分群算法少，聚敛的一跳覆盖区更少。由于时间紧迫，并没有完全实现，实为遗憾。

参考文献

[1] 运筹学教材编写组. 运筹学(修订版)[M]. 北京:清华大学出版社. 1990.3.
[2]曾庆黎，曾文艺.喷灌模型.数学的实践与认识[J]. 2004 年 4 月 34 期：16-22
[3]王晓东. 算法设计与分析 [M]. 北京：清华大学出版社.2003.1
[4]孙利民,李建中,陈渝等.无限传感器网络[M]. 北京：清华大学出版社. 2005.5
[5]郑相全等.无线自组网技术实用教程[M].北京：清华大学出版社.2004.6
[6] 最优化计算原理与算法程序设计. 粟塔山等著[M]. 长沙：国防科技大学出版社. 2002.12.
[7] 概率论与数理统计.（第二版）盛骤等著[M].北京：高等教育出版社.1989.8

附录：

MATLAB 程序：

```
clc;
clear all;
close all;
plus=sqrt(2)*100;
iniX=sqrt(2)*100/2;
iniY=sqrt(2)*100/2;
x0=zeros(1,10);
y0=zeros(1,10);
for i=2:10
    x0(1,i)=x0(1,i-1)+plus;
    y0(1,i)=y0(1,i-1)+plus;
end
[x0,y0]=meshgrid(x0,y0);
r=100;
plot_circle(r,x0+iniX+50,y0+iniY+50,'b') ;
hold on;
rectangle('Position', [300 350 1000 1000]);
%AXIS([0 1000 0 1000]);

clc;
clear all;
close all;
plus=sqrt(3)*100;
plus2=sqrt(3)*100/2;
plus3=3/2;
x=(1:20)*plus;
y=(1:20)*plus;
[x,y]=meshgrid(x,y);
x0=x;
y0=y;
for i=2:20
    x0(i,:)=x0(i-1,:)+plus2;
    y0(i,:)=y0(i-1,:)+plus3;
end
r=100;
figure,plot_circle(r,x0,y0,'b') ;
hold on;
% for i=0:5:5
        rectangle('Position', [500+1000 500 1000 1000]);
% end
```

C#程序：

```
        //求出邻接矩阵 dataA 和距离矩阵 dataAD
        private void neiberA()
        {
            int len = data.Length;
            for (int i = 0; i < len; i++)
            {
                float x0 = data[i][0];
                float y0 = data[i][1];
                dataA[i] = new short[len];
                dataA[i][i] = 1;//填充上三角矩阵
                dataAD[i] = new float[len];
                dataAD[i][i] = 0f;
                for (int j = i + 1; j < len; j++)
                {
                    float x1 = data[j][0];
                    float y1 = data[j][1];
                    float r = (float)Math.Sqrt((x1 - x0) * (x1 - x0) +
(y1 -  y0) * (y1 - y0));
                    dataAD[i][j] = r;
                    if (r > RR)
                    {
                        dataA[i][j] = 0;
                    }
                    else
                    {
                        dataA[i][j] = 1;
                    }
                    //求出最小距离
                    if (minD > r)
                        minD = r;
                }
            }
            //填充下三角矩阵
            for (int i = 0; i < len; i++)
            {
                for (int j = 0; j < i; j++)
                {
                    dataA[i][j] = dataA[j][i];
                    dataAD[i][j] = dataAD[j][i];
                }
            }
        }
```

```
        //自适应分群算法
        private void leastIDs()
        {
            int len = dataD.Length;
            int count = 0;
            bool [] tempData = new bool[len];
            //clusterIDs;
            for (int i = 0; i < len; i++)
            {
                tempData[i] = true;
            }

            for (int i = 0; i < len; i++)
            {
                if (tempData[i])
                {
                    tempData[i] = false;
                    int l = dataD[i].Length;
                    int temp_count = 0;
                    for(int j=1;j<l;j++)
                    {
                        if(tempData[dataD[i][j]])
                            temp_count++;
                    }
                    clusterIDs[count] = new int[temp_count + 1];
                    clusterIDs[count][0] = i;
                    temp_count = 0;
                    for (int j = 1; j < l; j++)
                    {
                        if(tempData[dataD[i][j]])
                        {
                            clusterIDs[count][++temp_count]          =
    dataD[i][j];
                            tempData[dataD[i][j]] = false;
                        }
                    }
                    count++;
                }
            }
            clusterIDs[count] = new int[1];
            clusterIDs[count][0] = -1;
        }
```