# 软件工程上机报告

# **Project 1**

姓名	学号	班级
蒋志远	U201517149	软工1506

## 问题描述

In a box bounded by [-1, 1], given \$m\$ balloons(they cannot overlap) with variable radio \$r\$ and position \$mu\$, find the optimal value of \$r\$ and mu which maximize sum \$r^2\$

## 问题分析

可以在方框的空白部分寻找最大的空白,然后用最大的圆进行填充,如图所示:

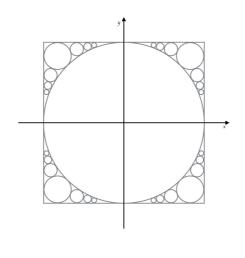


图1

我们按上述方案从大到小依次填满空隙,但是可能会出现下面的情况:

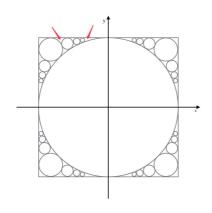


图2

箭头表明的部分右边的可能会出现比左箭头小的情况,这种情况由于计算过于复杂,所以我们 寻求次优解,将圆按图1的方式填满

图1由大到小的顺序顺序依次填,可以分为多种圆,各种圆的个数分别为\$1,4,8,8,8,\dots\$

第一个圆的半径为\$R*1=r*1=1\$,第二总圆的半径为\$R*2=r*2=r*3=r*4=r\_5=\frac{\sqrt{2}-1} {\sqrt{2}+1}\$

当圆的类别大于三时, 可以用一下公式计算半径

\$\$\left{

\begin{array}{1}

 $Rn = Veft( \frac{1-C(n-1)}{2(1+R(n-1))} \right)$ 

 $Cn = \sum_{i=2}^{n-1}2\sqrt{R}iR\{i-1\}+R2, n=3,4,5,\dots$ 

C2 = R2 \

 $r{5+8(n-3)+i} = Rn, i=1,2,\dots,8, n=3,4,5,\dots$ 

\end{array}

\right.

\$\$

#### 关键函数实现

```
Python
1
    def sub_solution_r(m):
2
         circles = []
         R = [1]
3
         circles.append(Circle(radius=1))
4
         sym_x = [1, 1, -1, -1]
5
         sym_y = [1, -1, 1, -1]
6
         if m == 0:
 7
             return []
8
         if m == 1:
9
             return circles
10
```

```
if m > 1:
1
             R1 = 3 - 2 * math.sqrt(2)
 2
 3
             R.append(R1)
            y = x = 1 - R1
4
             for i in range(0, 4):
5
                 circles.append(Circle((x * sym_x[i], y * sym_y[i]), R1))
6
                 if len(circles) == m:
 7
                     break
8
9
        if m > 5:
10
             pend_height = 0
11
             k = m - 5
12
13
             if k \% 8 == 0:
14
                 k = int(k / 8)
             else:
15
                 k = int(k / 8) + 1
16
             pend_current = R[1]
17
             for i in range(1, k+1):
18
                 r = ((1 - pend_current) / (2 * (1+math.sqrt(R[i])))) ** 2
19
                 pend_current += 2 * math.sqrt(r * R[i])
20
                 R.append(r)
21
                 x = 1 - r
22
                 y = 1 - pend_current
23
                 for j in range(4):
24
                     circles.append(Circle((x * sym_x[j], y * sym_y[j]), r))
25
                     if len(circles) == m:
26
                         break
27
                     circles.append(Circle((y * sym_y[j], x * sym_x[j]), r))
28
                     if len(circles) == m:
29
                         break
30
31
        return circles
```

## 测试用例:

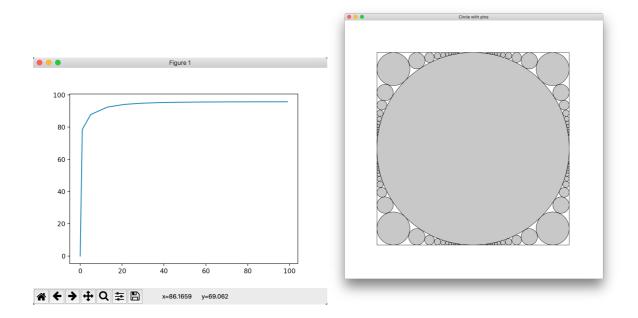
\$n\$	\$m\$	\$R_n\$
1	1	1
2	2~6	\$3-2\sqrt{2}\$
3	7 ~15	\$0.0857864376\dots\$
4	16 ~ 24	\$0.0513207883\dots\$

### 运行结果:

```
Center: (0.000000, 0.000000)
                                     Radius:1.0000000000
1
    Center: (0.828427, 0.828427)
2
                                     Radius: 0.1715728753
    Center: (0.828427, -0.828427)
3
                                     Radius: 0.1715728753
4
    Center: (-0.828427, 0.828427)
                                     Radius: 0.1715728753
    Center: (-0.828427, -0.828427)
                                     Radius: 0.1715728753
5
    Center: (0.914214, 0.414214)
                                     Radius: 0.0857864376
6
    Center: ( 0.414214, 0.914214)
7
                                     Radius: 0.0857864376
    Center: ( 0.914214, -0.414214)
                                     Radius: 0.0857864376
8
    Center: (-0.414214, 0.914214)
                                     Radius: 0.0857864376
9
    Center: (-0.914214, 0.414214)
                                     Radius: 0.0857864376
10
    Center: ( 0.414214, -0.914214)
                                     Radius: 0.0857864376
11
    Center: (-0.914214, -0.414214)
                                     Radius: 0.0857864376
12
    Center: (-0.414214, -0.914214)
                                     Radius: 0.0857864376
13
    Center: (0.948679, 0.281509)
                                     Radius: 0.0513207883
14
    Center: (0.281509, 0.948679)
                                     Radius: 0.0513207883
15
    Center: (0.948679, -0.281509)
                                     Radius: 0.0513207883
16
    Center: (-0.281509, 0.948679)
                                     Radius: 0.0513207883
17
    Center: (-0.948679, 0.281509)
                                     Radius: 0.0513207883
18
    Center: (0.281509, -0.948679)
19
                                     Radius:0.0513207883
    Center: (-0.948679, -0.281509)
                                     Radius: 0.0513207883
20
21
    Center: (-0.281509, -0.948679)
                                     Radius: 0.0513207883
    Center: (0.965886, 0.197825)
                                     Radius: 0.0341137321
22
    Center: (0.197825, 0.965886)
23
                                     Radius: 0.0341137321
    Center: ( 0.965886, -0.197825)
                                     Radius: 0.0341137321
24
25
    Center: (-0.197825, 0.965886)
                                     Radius: 0.0341137321
    Center: (-0.965886, 0.197825)
                                     Radius:0.0341137321
26
27
    Center: (0.197825, -0.965886)
                                     Radius: 0.0341137321
    Center: (-0.965886, -0.197825)
                                     Radius: 0.0341137321
28
    Center: (-0.197825, -0.965886)
                                     Radius: 0.0341137321
29
    Center: (0.975694, 0.140235)
                                     Radius: 0.0243059818
30
31
```

#### 结论

当m从0增长到99时,计算每个m值下的覆盖率,得出一下函数图像:



从图中可以看出,用此种方案填放并不能达到最优,当m达到一定大小时,对覆盖率的提供并不大,空缺的部分就是图2中类似左箭头的部分已经成为主要的空缺部分

# 附录

Gitlog

```
Git
```

commit 9e9c441469f16ee13c8797c1fb0817bc846a0eb2 1 Author: BluesJiang <763400095@gg.com> 2 Thu Jun 1 11:43:50 2017 +0800 Date: 3 4 fix when m < 3, result goes wrong 5 6 commit 834494816cb0729923c505ecef89a2ee0131e7bd 7 Author: BluesJiang <763400095@qq.com> 8 Thu Jun 1 10:56:47 2017 +0800 Date: 9 10 bug fixed 11 12 commit 017a074fc3b1b28dc07fd1314ae2fb13aa485ecd 13 Author: BluesJiang <763400095@qq.com> 14 Thu Jun 1 10:30:55 2017 +0800 15 Date: 16 clear the formula 17 18 commit c851ac95780a02f307bbac35886900cb99cc4af2 19 Author: BluesJiang <763400095@qq.com> 20 Fri May 5 20:45:27 2017 +0800 21 22 23 sub\_solution