按面积布点算法开发文档

一、 背景介绍

给出一个房间的可布置区域、探测区域和房间框线，以及需要布置探测器的探测半径和探测面积，在房间内布置探测器。要求探测范围完全覆盖探测区域，布置探测器数量尽可能少，布局尽量横平竖直。

二、 算法接口：

输入：

List<polygon> area_room;//房间框线

List<polygon> area_detect;//探测区域

List<polygon> area_layout;//可布置区域

bool IsLayoutByBeams;//是否按照避梁进行布置

double protect_area;//保护面积

double protect_radius;//保护半径

double MinStep;//最小间距（可选参数）

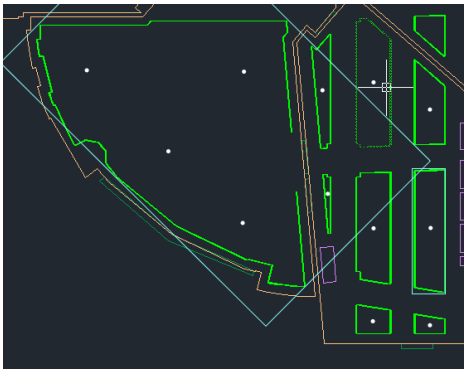
输出：

List<Point3d> points;//房间点位布置

三、 算法思路：

3.1 按梁布置

1、 计算每个可布置区域的中心和最小外接矩形，沿最小外接矩形的方向以一合适的间距设置布置点的可能位置

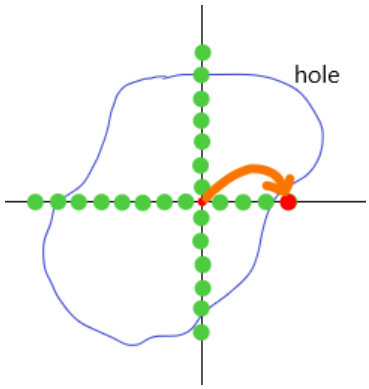


a) 外接矩形的长宽都小于 dist，则直接设置为该区域的中心

b) 外接矩形某一方向的长度 len 大于 dist，则在该方向均匀布置[len/dist]个点，

i. 若某个点的位置不在可布置区域外框线内，舍去

ii. 若某个点在可布置区域的洞内，按照下图的方式移动点位，其中坐标轴方向由可布置区域外接矩形确定



c) 输出形式：n_{all} 个点的坐标集合 position_list_

2、 构建点、可布置区域、探测区域的索引字典，封装成 room 类，将点按照从属的探测区域进行分组

3、 计算完全覆盖区域 C 所需要的探测器数量范围

a) Min=max{[$S_{覆盖区域}/S_{探测面积}$]+1，探测区域个数}

b) Max=min+n_{set}，其中 n_{set} 为预估最多点数

4、 染色体形式：

a) 布置探测器个数 n_i，范围是第三步计算的范围

b) 长度为 n_{all} 的二进制信息，n_{all} 为第一步中布置的点的总数，若第 j 位为 1，则表示在 Position[j]处安放探测器

c) 安放探测器的位置集合 real_pos_list_

d) 初始化时先在每个探测区域内随机布一个点，剩余点再随机布置

5、 适应度函数：

f=p1*覆盖率+p2*ni +p3*共线个数+p4*均匀度

a) 覆盖率的计算：对 ni 个圆进行求并，最后与区域 C 求交集 R_cover，覆盖率=S_{Cover}/S_C

b) 共线个数：每次寻找经过点数最多的直线，直到所有点都在直线集上，共线个数是最少直线数

c) 均匀度：按照 x，y 方向计算点的分布律，之后计算与均匀分布之间的 K-L 散度，K-L 散度越小越均匀

d) p1 为 k*覆盖率^2，k 值较大，当覆盖率不同时，p1 起主导作用

e) p2 为负值，用于确保覆盖率接近时，使用的探测器数量越大，效果越差

f) p3、p4 用于美观度的判断，均为负数

6、 选择：按照适应度轮盘进行，先计算适应度总数，适应度为 fitvalue 的染色体存活的概率是 fit_value/fit_sum

7、 交叉：从上一代的染色体中寻找两条染色体，一条是爸爸，一条是妈妈，将这两条染色体按照探测区域进行切断（适应度高的染色体在组内的布点位置应当是比较合理的），子染色体按组随机选择父亲或者母亲的基因段

8、 变异：

a) 第一种变异形式：染色体的二进制段上随机选择 1 个位进行修改（点数增加/减少）

b) 第二种变异形式：分别从二进制段的值为 0 的位和值为 1 的位中随机选取 m 个位进行修改（m 个点进行了重新布置）

c) 第三种变异形式：对于染色体二进制位为 1 的位置随机进行小范围移动，即更新探测器的位置

8、迭代总数：直接给定

遗传算法总体框架：

1、 初始化变量

a) 种群数量 popsize
b) 迭代次数 iteration
c) 选择概率 ps
d) 交叉概率 pc
e) 变异概率 pm1,pm2,pm3

2、 初始种群

```
pop = initpop(popsize);
```

3、遗传进化

```
for (i =1;i<=iteration;i++){
    //计算适应度值（函数值）
    objvalue = cal_objvalue(pop);
    //选择操作
    newpop = selection(pop,ps);
    //交叉操作
    newpop = crossover(newpop,pc);
    //变异操作：随机概率 rand 在(0,pm1)中按照第一种方式变异，在(pm1,pm2)中按
    //照第二种方式变异，在（pm2，pm3）中按照第三种方式变异，剩余不变异
    newpop = mutation(newpop,pm1,pm2,pm3);
    //更新种群
    pop = newpop;
}
//寻找最优解
[bestindividual,bestfit] = best(pop,fitvalue);
```