# **2020美赛A题(3)-元胞自动机模拟随机游走**

[](https://www.zhihu.com/people/hu-meng-qi-41-5)

**[isle](https://www.zhihu.com/people/hu-meng-qi-41-5" \t "https://zhuanlan.zhihu.com/p/_blank)**

​关注他

4 人赞同了该文章

今天才肝完校赛，想到建模的内容已经很久没更了。后面如果有时间可能会把这次校赛的内容搬上来，因为这次我们组选择的是图像处理的题目，可以说这个方向我们之前从来没有接触过，在这里记下来或许对之后的建模比赛有好处。

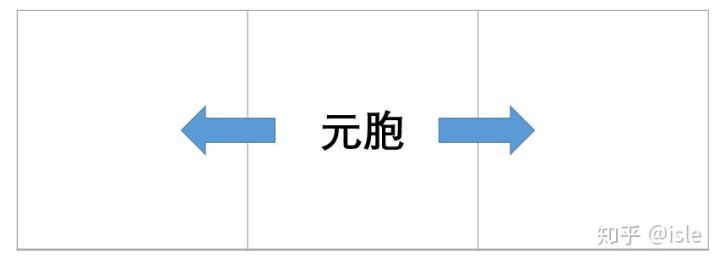
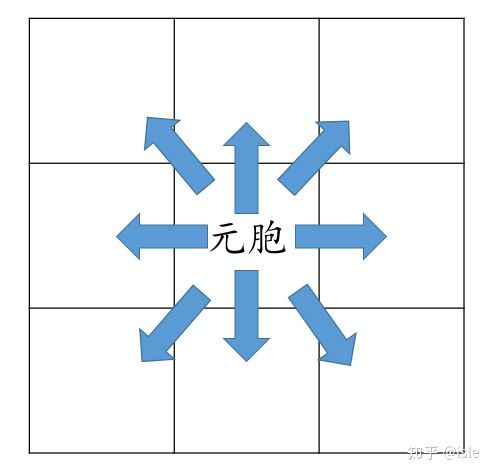
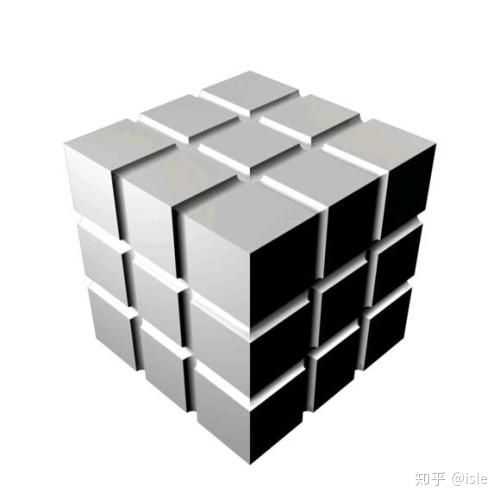
这篇文章要完成的是美赛A题第一问的第三个部分用元胞自动机来模拟两种鱼的游动，这个游动跟元胞自动机的一个重要应用——随机游走（random walk）非常相关。下面是这篇文章的主要内容：

* 介绍元胞自动机模拟随机游走
* 用一个例题来说明模拟过程
* 说明美赛A题的实现过程

## **介绍元胞自动机模拟随机游走**

**随机游走（random walk）**  
也称随机漫步，随机行走等是指基于过去的表现，无法预测将来的发展步骤和方向。核心概念是指任何无规则行走者所带的守恒量都各自对应着一个扩散运输定律 ，接近于布朗运动，是布朗运动理想的数学状态，现阶段主要应用于互联网链接分析及金融股票市场中。

根据随机游走的定义和性质可以发现，元胞自动机仿真算法是一个可以模拟随机游走的利器。例如：在一维直线的随机游走可以模拟为元胞向左右两个方向的等可能转移；在二维平面的随机游走可以模拟为元胞向八个方向的等可能转移；在三维空间内的随机游走可以模拟为元胞向二十六个方向的等可能转移。如下三个图所示：

一维直线的情况二维平面的情况三维空间的情况

三维图就只能意会一下，元胞是所有27个立方体的最中心的那个，一共有26个方向可以进行转移。

## **一个例题**

在一个封闭的空房间内，一个扫地机器人在里面随机移动，房间某个墙边的位置有充电插座  
1.扫地机器人是否一定会找到充电插座？仿真计算机器人移动到充电插座的可能性和所需要的平均时间，与房间大小的关系是什么？  
2.扫地机器人是否能清扫到这个封闭房间的所有位置？如果可以，清扫时间和房间大小的关系是什么？

**该问题的假设**

* 当扫地机器人在房间的内部时，在一个时间间隔后它可以向周围八个方向行走。
* 当扫地机器人在房间的边界时，在一个时间间隔后它只能向除墙壁以外的五个方向行走
* 当扫地机器人在房间的角落时，在一个时间间隔后它只能向除角落的两个墙壁以外的三个方向行走
* 房间的大小为m\*n（在本文中以22\*18为例）
* 第一行、第一列、最后一行和最后一列设定为墙壁
* 充电插头设定在（1，1）的位置（这里也可以自行设置，可以看看设定在边界和角落的不同）
* 扫地机器人的初始位置是在房间中随机生成的（这里也可以看看设定不同初始位置的扫地机器人会对结果有什么影响）

**模型的建立**

状态定义

IMG_260

红色表示扫地机器人已经经过的位置，且其对应的RGB矩阵的坐标为： IMG_261

白色表示没有扫地机器人经过的位置，且其对应的RGB矩阵的坐标为： IMG_262

黑色表示墙壁的位置，且其对应的RGB矩阵的坐标为： IMG_263

状态转移概率矩阵

当机器人在房间内部时：

1. 周围八个位置有机器人，则矩阵为： IMG_264
2. 周围八个位置没有机器人，则矩阵为： IMG_265

当机器人在房间边界时：

1. 周围五个位置有机器人，则矩阵为： IMG_266
2. 周围五个位置没有机器人，则矩阵为：IMG_267

当机器人在房间角落时：

1. 周围三个位置有机器人，则矩阵为： IMG_268
2. 周围三个位置没有机器人，则矩阵为：IMG_269

实现代码

遍历房间所有位置的代码：

*%一间封闭的空房间, 一个人在里面随意移动%状态变化规则：%1.四周没有墙壁时，以等概率方式移动到周围8个位置之一；%2.位于墙角时，以等概率方式移动到周围3个位置之一；%3.位于墙边非墙角时，以等概率方式移动到周围5个位置之一*close all;clc;clear;M=22;*%房间大小M行,包括两行墙*N=18;*%房间大小N列,包括两列墙%房间状态初始化*R=ones(M,N);*%初始化R房间矩阵*R(1,:)=0;R(M,:)=0;R(:,1)=0;R(:,N)=0;*%R的边界初始化为0*G=R;*%G初始化为R*B=R;*%B初始化为R*i=unidrnd(M**-**2)**+**1;*%随机选取初始位置行标*j=unidrnd(N**-**2)**+**1;*%随机选取初始位置列标*G(i,j)=0;*%在G中更新初始点*B(i,j)=0;*%在B中更新初始点*ima=cat(3,R,G,B);*%ima为R，G，B拼接而成的3维数组*imshow(ima,'InitialMagnification','fit');*%初始状态图显示*

neighbour=[**-**1 **-**1;**-**1 0;**-**1 1;0 **-**1;0 1;1 **-**1;1 0;1 1];*%当前位置邻域%i,j始终表示人所在位置的坐标*t=0;*%初始化时间***while** 1

t=t**+**1;*%更新时间*

**if** 2**<**i **&&** i**<**M**-**1 **&&** 2**<**j **&&** j**<**N**-**1 *%人在内部*

r=unidrnd(8);

**elseif** i**==**2 **&&** j**==**2 *%人在左上墙角*

r=randsrc(1,1,[5 7 8]);

**elseif** i**==**2 **&&** j**==**N**-**1 *%人在右上墙角*

r=randsrc(1,1,[4 6 7]);

**elseif** i**==**M**-**1 **&&** j**==**2 *%人在左下墙角*

r=randsrc(1,1,[2 3 5]);

**elseif** i**==**M**-**1 **&&** j**==**N**-**1 *%人在右下墙角*

r=randsrc(1,1,[1 2 4]);

**elseif** i**==**2 **&&** 2**<**j **&&** j**<**N**-**1 *%人在上墙边非墙角*

r=randsrc(1,1,[4 5 6 7 8]);

**elseif** i**==**M**-**1 **&&** 2**<**j **&&** j**<**N**-**1 *%人在下墙边非墙角*

r=randsrc(1,1,[1 2 3 4 5]);

**elseif** 2**<**i **&&** i**<**M**-**1 **&&** j**==**2 *%人在左墙边非墙角*

r=randsrc(1,1,[2 3 5 7 8]);

**else** 2**<**i **&&** i**<**M**-**1 **&&** j**==**N**-**1; *%人在右墙边非墙角*

r=randsrc(1,1,[1 2 4 6 7]);

**end**

i=i**+**neighbour(r,1); *%更新位置坐标*

j=j**+**neighbour(r,2); *%更新位置坐标*

G(i,j)=0; *%更新当前人所在位置为红色*

B(i,j)=0; *%更新当前人所在位置为红色*

ima=cat(3,R,G,B);*%更新ima数组*

imshow(ima,'InitialMagnification','fit');

sum=0;*%初始化sum*

**for** m=1:M

**for** n=1:N

sum=sum**+**G(m,n); *%计算G矩阵的各个元素之和*

**end**

**end**

**if** sum**<**1

**break**;

**end**

*%如果元素之和恰好等于0，就说明遍历了所有的位置*

pause(0.0001);*%间隔0.0001秒***end**

找到充电插座的代码：

*%一间封闭的空房间, 一个人在里面随意移动%状态变化规则：%1.四周没有墙壁时，以等概率方式移动到周围8个位置之一；%2.位于墙角时，以等概率方式移动到周围3个位置之一；%3.位于墙边非墙角时，以等概率方式移动到周围5个位置之一*close all;clc;clear;M=22;*%房间大小M行,包括两行墙*N=18;*%房间大小N列,包括两列墙%房间状态初始化*R=ones(M,N);*%初始化R房间矩阵*R(1,:)=0;R(M,:)=0;R(:,1)=0;R(:,N)=0;*%R的边界初始化为0*G=R;*%G初始化为R*B=R;*%B初始化为R*i=unidrnd(M**-**2)**+**1;*%随机选取初始位置行标*j=unidrnd(N**-**2)**+**1;*%随机选取初始位置列标*G(i,j)=0;*%在G中更新初始点*B(i,j)=0;*%在B中更新初始点*ima=cat(3,R,G,B);*%ima为R，G，B拼接而成的3维数组*imshow(ima,'InitialMagnification','fit');*%初始状态图显示*

neighbour=[**-**1 **-**1;**-**1 0;**-**1 1;0 **-**1;0 1;1 **-**1;1 0;1 1];*%当前位置邻域%i,j始终表示人所在位置的坐标*t=0;*%初始化时间***while** 1

t=t**+**1;*%更新时间*

**if** 2**<**i **&&** i**<**M**-**1 **&&** 2**<**j **&&** j**<**N**-**1 *%人在内部*

r=unidrnd(8);

**elseif** i**==**2 **&&** j**==**2 *%人在左上墙角*

r=randsrc(1,1,[5 7 8]);

**elseif** i**==**2 **&&** j**==**N**-**1 *%人在右上墙角*

r=randsrc(1,1,[4 6 7]);

**elseif** i**==**M**-**1 **&&** j**==**2 *%人在左下墙角*

r=randsrc(1,1,[2 3 5]);

**elseif** i**==**M**-**1 **&&** j**==**N**-**1 *%人在右下墙角*

r=randsrc(1,1,[1 2 4]);

**elseif** i**==**2 **&&** 2**<**j **&&** j**<**N**-**1 *%人在上墙边非墙角*

r=randsrc(1,1,[4 5 6 7 8]);

**elseif** i**==**M**-**1 **&&** 2**<**j **&&** j**<**N**-**1 *%人在下墙边非墙角*

r=randsrc(1,1,[1 2 3 4 5]);

**elseif** 2**<**i **&&** i**<**M**-**1 **&&** j**==**2 *%人在左墙边非墙角*

r=randsrc(1,1,[2 3 5 7 8]);

**else** 2**<**i **&&** i**<**M**-**1 **&&** j**==**N**-**1; *%人在右墙边非墙角*

r=randsrc(1,1,[1 2 4 6 7]);

**end**

i=i**+**neighbour(r,1); *%更新位置坐标*

j=j**+**neighbour(r,2); *%更新位置坐标*

G(i,j)=0; *%更新当前人所在位置为红色*

B(i,j)=0; *%更新当前人所在位置为红色*

ima=cat(3,R,G,B);*%更新ima数组*

imshow(ima,'InitialMagnification','fit');

**if** i**==**2 **&&** j**==**2

**break**;

**end**

pause(0.0001);*%间隔0.0001秒***end**

其实这道题还远没有做完，有兴趣的读者可以自己验证一波题目中的几个问题。（其实是我想偷个懒。。）

## **美赛A题的实现**

**题目假设**

* 鲱鱼的游动速度是鲭鱼的两倍

由联合国粮农组织的数据[[1]](https://zhuanlan.zhihu.com/p/139787976" \l "ref_1)可知，鲱鱼的游动速度约为鲭鱼游动速度的两倍，则在元胞自动机的体现为，同一个时间间隔鲱鱼可以移动两格，而鲭鱼只能移动一格。

* 为简化模型，鱼群在迁徙过程中只受海洋温度的影响。
* 选取41\*81的区域作为实现元胞自动机的区域
* 两种鱼类会向适合生存的水温（9℃-12℃）的方向游动，如果存在两个或以上的位置适合生存，则随机走向其中一个

由于大西洋水温的改变会使得某些区域的水温不再适合两种鱼类的生存，而同时另一些地方又开始适合鱼类生存，所以由此产生迁徙。

* 每一个时间间隔为一年

这是由于收集到全球的海温数据是以年为单位所以我们设定的时间间隔为一年。

**模型建立**

状态定义

IMG_270

状态转移规则

* 如果鱼周围八个格点中没有格点温度在9℃和12℃之间，那么鱼不移动。
* 如果鱼周围的八个格点中只有一个格点温度在9℃和12℃之间那么鱼会向这个格点移动。
* 如果鱼周围的八个格点中有两个或者以上的格点温度在9℃和12℃之间那么鱼会在这几个格点中随机选择一个来移动。

状态转移概率矩阵

如果鱼周围在9℃和12℃之间的格点有N个 IMG_271

1. 周围八个位置有鱼，则矩阵为： IMG_272
2. 周围八个位置没有鱼，则矩阵为： IMG_273

**实现代码**

load("20**-**69预测海温.mat");*%我们选择原始数据中的10-50行和140-220列*Fa=zeros(41,81);Fb=zeros(41,81);*%初始化中间矩阵1，2*R=zeros(41,81);G=zeros(41,81);B=ones(41,81);B=B**./**2;*%初始化RGB矩阵*C=zeros(41,81,3);*%初始化绘图矩阵C*set(gcf,'DoubleBuffer','on');*%设定图片适应屏幕大小*ox=30;oy=25;*%给出鱼的初始位置*M=min(T1);*%选择出陆地的位置*D=zeros(2,51);*%产生一个空矩阵来表示鱼的位置*D(:,1)=[ox,oy];*%记录鱼的初始位置*n=0;*%初始化n*title(['T = ',num2str(n)]);*%在标题处显示时间***while** n**<**50

n=n**+**1;*%每次更新都要使n加1，可以理解为时间单位加1*

F=T1((ox**-**1):(ox**+**1),(oy**-**1):(oy**+**1),n);*%选取鱼附近的3\*3的格点位置*

N=length(find(F**<**1200**&**F**>**900)); *%记录在指定温度内的格点的个数*

**if** N**>**0 *%如果满足条件的格点数大于零*

[r,c]=find(F**<**1200**&**F**>**900); *%输出这些格点的位置*

m=unidrnd(N);*%随机选择格点*

ox=ox**-**2**+**r(m);

oy=oy**-**2**+**c(m);*%更新鱼的位置*

D(:,n**+**1)=[ox,oy];*%在D中记录鱼的位置*

**else***%如果没有满足要求的格点，那么位置不变*

ox=ox;

oy=oy;

**end**

**for** i=1:41

**for** j=1:81

**if** T1(i,j,n)**>**0

Fa(i,j)=T1(i,j,n);

**else**

Fb(i,j)=T1(i,j,n);

**end**

**endend***%将原始数据的第一层导入矩阵Fa，Fb%规则是将原始数据中正整数的部分导入Fa%将负整数的部分导入Fb***for** i=1:41

**for** j=1:81

**if** T1(i,j,n)**==**M

Fb(i,j)=0;

G(i,j)=1;

**end**

**endend***%这一个循环的意思是将数据中陆地的位置标注出来%将Fb中最小的值全部转换为0%G显色矩阵中的相应位置标注为1*

R=Fa**./**max(Fa);*%将矩阵Ea归一化赋给R显色矩阵*

G=G**+**Fb**./**min(Fb)**.\***0.9;G(ox,oy)=1;

*%将矩阵Eb归一化并且乘以0.9再累加G显色矩阵*

*%这里要说明一下0.9的作用，本质上是为了调节图形可视化后的颜色*

C(:,:,1)=R;

C(:,:,2)=G;

C(:,:,3)=B;

*%将RGB显色矩阵赋给C*imshow(C,'InitialMagnification','fit');*%显示图像*hold on*%保持图像*x = 0.5:81.5; *%假设水平分成81格*y = 0.5:41.5; *%假设垂直分成41格*

M = meshgrid(x,y); *%产生网格*

N = meshgrid(y,x); *%产生网格*

hold on*%保持图像*

plot(x,N,'k'); *%画出水平横线*

plot(M,y,'k'); *%画出垂直竖线*

title(['T = ',num2str(n)]);*%第二行是设置图像的标题，显示T=当前时刻*

pause(0.01);*%暂停0.1s*

f=getframe(gcf);

imind=frame2im(f);

[imind,cm] = rgb2ind(imind,256);

**if** n **==** 1

imwrite(imind,cm,'infection1.gif','GIF', 'Loopcount',inf,'DelayTime',0.01);

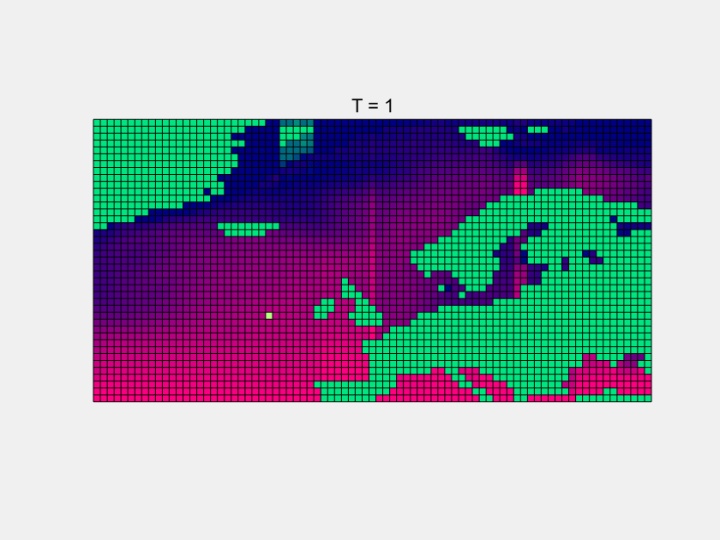
**else**

imwrite(imind,cm,'infection1.gif','GIF','WriteMode','append','DelayTime',0.01);

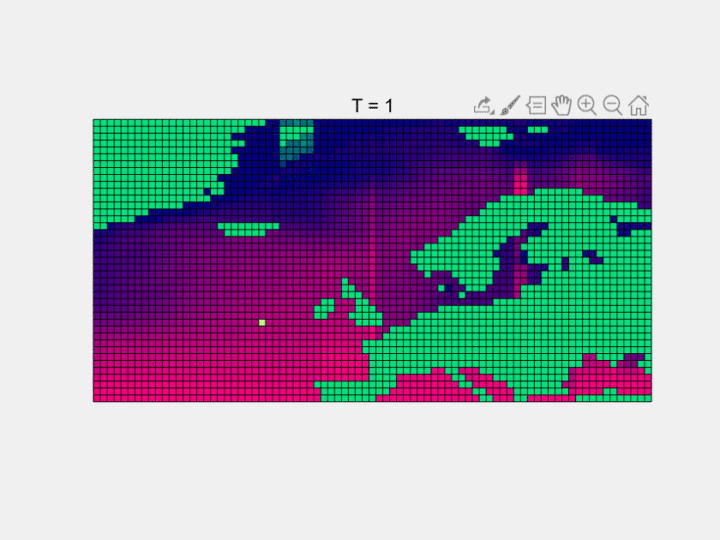
**end** **end**

上述是鲭鱼的游动代码。同时鲱鱼的游动代码和鲭鱼的游动代码非常类似，只用把移动的过程多增加一次即可，这里就不贴上来了。有兴趣的读者可以在评论区留言讨论。

**实现结果**



鲭鱼游走



鲱鱼游走

同时在这个代码中也可以记录鲱鱼和鲭鱼行走的路线，为之后的计算经济效应做铺垫。

## **结语**

用三篇文章才最终把2020美赛A题的第一问讲完，看似非常复杂其实如果掌握了其中的原理也就是稍微改变一下现有的代码就可以完成的事情。对于任意一个建模比赛来说，前期的积累是一个方面，**真正的核心能力在我看来是在短时间内（三天？四天？）学习新知识的能力。**每一道题都是不同的，要具体问题具体分析，适当改变算法和代码。