程序模拟基于二进制逻辑的细胞群体行为

完成人

黄寅殊

指导小组成员

蔡亮 研究员

目　录

摘　要 I

Abstract II

一、前言 1

1.1 实验得到的细胞增殖规律 1

1.2 实验体系研究多细胞行为的简介和进展 1

1.3 各种细胞自动机介绍 2

1.3.1 初等细胞自动机 2

1.3.2 生命游戏（**Game of Life**） 3

1.3.3 神经干细胞生长动力学模型 3

1.4 人工智能和数值模拟的发展及前景 4

1.4.1 人工生命与人工智能 4

1.4.2 人工生命研究进展 5

1.4.3 人工生命发展前景 6

二、材料与方法 7

2.1 细胞自动机 7

2.2 实验方法 9

三、研究结果 11

3.1 程序模拟干细胞增殖 11

3.2 程序模拟两种细胞增殖 13

3.3 模拟两种细胞相互作用 15

3.3.1 促进分裂 15

3.3.2 抑制分裂.....16
3.3.3 促进凋亡.....17
3.3.4 抑制凋亡.....18
四、讨 论.....20
4.1 微观规律分析.....20
4.2 宏观规律分析.....20
4.3 改进方法.....21
4.4 结论.....21
参考文献.....23
致谢.....24
附录.....25

摘　要

细胞群体行为是指由多个相互影响、相互作用、相互依赖的细胞组成的集合体的相对运动现象。细胞之间的相互作用可以有效地调节和影响细胞的群体行为，如细胞分化，又如癌细胞和免疫细胞的相互作用。细胞自动机是研究细胞群体行为的有效途径，不仅能模拟单种细胞的增殖过程，还能模拟多种细胞间的相互作用。本研究参考了目前使用较为广泛的几种细胞自动机模型，构建了一个基于二进制逻辑的细胞自动机。实验结果表明，本研究所建立的细胞自动机模型可以很好地模拟单种及两种不同细胞增殖、细胞间抑制和促进增殖等几种细胞群体行为，验证了其微观及宏观规律，有助于深入研究细胞相互作用的宏观意义。本工作的代码及测试结果在 https://github.com/left-fdu/cellautomata。

关键词：细胞自动机，细胞群体行为，人工生命

Abstract

Cell social behavior refers to the movement of aggregates composed of multiple interacting and interdependent cells. The interactions between cells can effectively regulate and influence cell social behavior, such as cell differentiation, as well as the interactions of cancer cells and immune cells. It is of great significance to study cell social behavior for further understanding the interactions between cells. Cellular automata can be effectively used in the study of cell social behavior. It can not only simulate the proliferation process of single cell, but also simulate the interactions between multiple cells. Inspired by several widely used cellular automata models, this paper constructs a cellular automata based on binary logic. The cellular automata can simulate the behavior of various cell groups and quantifies their microscopic and macroscopic behaviors. Our analysis provides cell social behavior, such as proliferation of two different kinds of cells, intercellular inhibition and promotion of proliferation between cells, and successfully demonstrate the value using computer simulation to obtain macroscopic understanding of molecular interactions between cells. For the work code and test results, please refer to my reusable toolkit cell automata at https://github.com/left-fdu/cellautomata.

Key words: Cellular automata, Cell social behavior, Artificial life

一、前言

本研究的目的是通过计算机程序模拟细胞自我复制、竞争等群体行为。研究发现细胞群体行为不仅仅依赖于单个细胞水平的动力学，还受到多个细胞之间相互作用的影响，这些行为中的一些对细胞集落的进化有着关键的影响。本研究拟通过编写特定程序，研究者通过该程序可以设定不同的初始参数，调整这些关键因素，并观察模拟的细胞群体行为，验证其内在规律。

1.1 实验得到的细胞增殖规律

细胞是生命活动的基本单位。细胞的增殖、分化和凋亡依赖细胞的自我调节和细胞外信号的调节。如果细胞增殖、分化和凋亡的过程是无序的，它通常意味着严重的疾病。

细胞可根据分裂、分化的能力分为干细胞和正常细胞。干细胞可分为全能干细胞、多能干细胞和专能干细胞。干细胞具有很强的分裂和分化能力，全能干细胞具有发育成完整个体的潜能，专能干细胞可以分化为特定的功能体细胞。另外，肿瘤细胞具有无限增殖的能力。

对普通细胞而言，单种细胞自我增殖是有限度的。当多细胞生物的体细胞在体外培养时，它们遵循贴壁生长和接触抑制的规则。肿瘤细胞能够无限增殖，细胞周期短，繁殖速度快，且细胞接触对癌细胞的增殖无抑制作用。

细胞间的信息交换对细胞群体的行为也至关重要。细胞间的信息交换通过各种信号分子进行，如多肽、酰基高丝氨酸内酯等。细胞通过识别信号来调节自身的行为。例如，Notch 蛋白与其受体结合会激活巨噬细胞中的信号，从而维持干细胞的浓度。

1.2 实验体系研究多细胞行为的简介和进展

细胞自动机是研究细胞群体行为的有效途径，并且已经经历了数十年的发展。

在 20 世纪 50 年代早期，计算机之父冯·诺伊曼提出细胞自动机的概念来模拟生命系统的自我复制功能。经过众多数学家、物理学家的深入研究，细胞自动

机理论发展迅速。在物理学方面，细胞自动机是一种离散动力学系统。该系统按照一定的局部规则，在一个由具有离散、有限状态的细胞组成的细胞空间上，在离散的模拟次数维度上演化。

生物学家将细胞自动机视为生命活动的抽象。细胞自动机可以通过计算机建模和仿真的方法研究细胞群体行为，即研究生物细胞的大量并行单元个体组成的复杂系统的宏观行为，以及其内在的规律。

细胞自动机的设计思想源于生物自我传播的思想，广泛用于肿瘤细胞的增长机理模拟、艾滋病病毒 HIV 的感染过程、细胞自我组织、自我繁殖、克隆等生命活动的研究。另外，细胞自动机和神经网络、遗传算法等人工生命方法一样，都是基于局部的相互作用，来研究系统的整体行为。它们相互结合也产生了新的交叉模型。

1.3 各种细胞自动机介绍

1.3.1 初等细胞自动机

初等细胞自动机(Elementary Cellular Automata，简称 ECA)是每个细胞只有两种状态、两个邻居的一维细胞自动机。状态集 S 中的元素符号可取 {0，1}，{-l，1}，{生，死}等等。一种简单的映射规则如下：

t:	111	110	101	100	011	010	001	000
t+1:	0	1	0	0	1	1	0	0

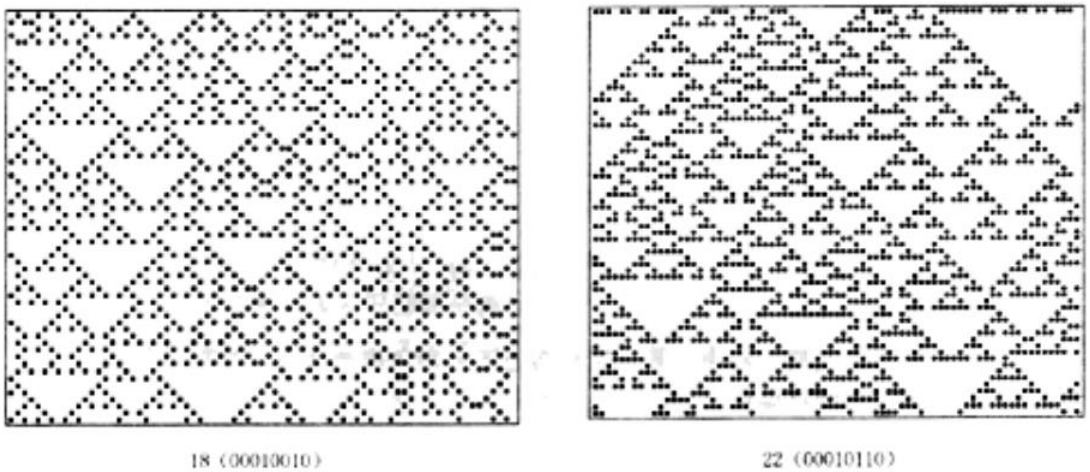


图 1-1 几种初等细胞自动机（周成虎，1999）

黑色方块代表 1，白色方块代表 0；图中纵向坐标为模拟次数，反映一维细胞自动机随模拟次数的演化

对于任何一个一维的二进制序列，都只有这八种映射，这样的组合总共有 2^8 = 256 种，即初等细胞自动机只有 256 种不同规则。

1.3.2 生命游戏（Game of Life）

Game of Life 是 John Howden Conway 在 20 世纪 60 年代末设计的单人电脑游戏。与围棋类似，生命游戏中的细胞有两种状态：{生,死}。细胞分布在规则划分的网格上，以相邻的八个细胞为邻居；细胞的生死取决于它自己的生死状态和周围八个邻居的状态。

如果一个细胞是“生”，且八个相邻细胞中有两个或三个为“生”，则细胞继续存活，否则它会死亡；如果一个细胞为“死”，且八个相邻细胞中正好有三个为“生”，则该细胞将会复活，否则保持为死的状态。

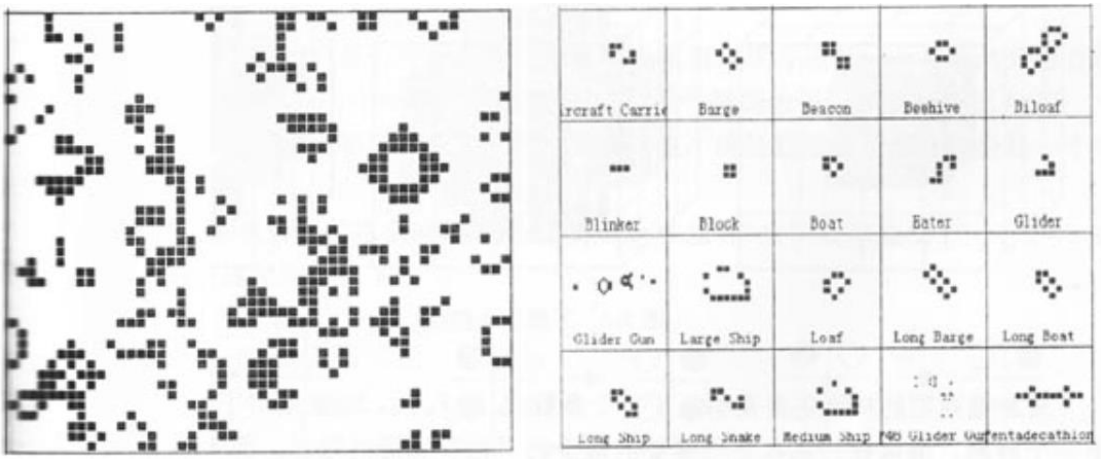


图 1-2 生命游戏及若干常见的图案（周成虎，1999）

生命游戏能够产生丰富的动态图案和动态结构。随着人们对“生命游戏”研究的深入，出现了许多生命游戏的变种和扩展，例如三维、四维空间的生命游戏。

1.3.3 神经干细胞生长动力学模型

该模型是在三维层面模拟神经干细胞球的分裂、生长、分化过程，根据神经干细胞球内各种细胞成分的实际培养结果，制定具体的分化、分裂规则。

在该模型中，每个细胞以等边三角形接触排列，每个细胞有 12 个邻居。当干细胞成熟，且周围有足够空间时，它会分裂成两个子代干细胞。新生细胞需要模拟一定次数生长成为成熟的细胞。按照分裂原则，干细胞第三次分裂会变成两个前体细胞。前体细胞第一次分裂变成两个前体细胞，第二次则分裂成一个前体细胞和一个程序性死亡细胞。

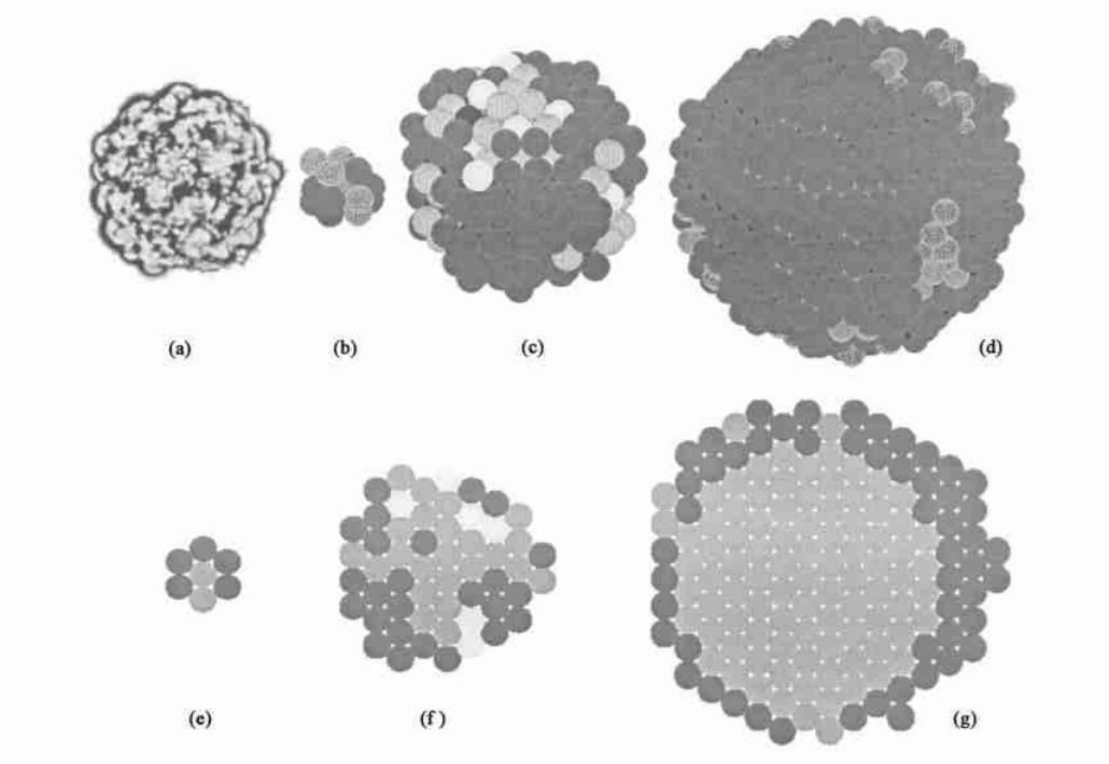


图 1-3 静态培养神经干细胞球生长过程的典型模拟结果（刘天庆，2006）
（a）神经球实际照片，（b）（c）（d）为分别培养 32h、80h、120h 的神经球 3D 图；（e）（f）（g）分别对应于的切面图

此三维模型考虑了细胞分裂、分化、细胞接触抑制、不同种类的细胞相互作用，并且可以计算神经干细胞浓度。

1.4 人工智能和数值模拟的发展及前景

1.4.1 人工生命与人工智能

人工生命是一门基于“生物学程序模拟”的理念形成的新兴学科 。它试图在人工媒体（如计算机）上模拟与生物有机体相关的基本活动，如自我复制、寄生、竞争等。通过研究和观察“可能的生命现象”（Life-as-it-could-be），从而加深人们对“已知的生命现象”（Life-as-we-know-it）的理解。人工智能则是对人类大脑思维活动的模拟。

人工生命、人工智能的理论发展和虚拟现实等技术的发展给研究者提供了新的研究思路。人工生命方法通过一些简单的规则可以产生复杂的行为模式。我们可以通过程序在较短模拟次数内模拟生物进化历程，也可以通过虚拟生物研究生命的机理。

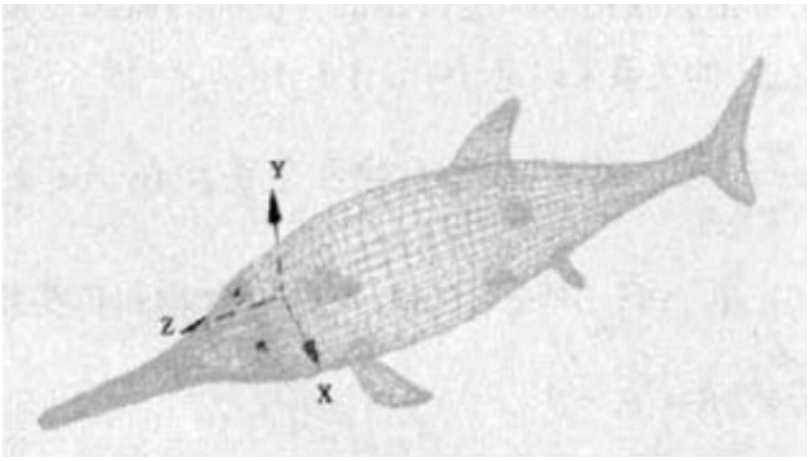


图 1-4 人工生命模拟的应用（班晓娟，2009）
人工生命方法建模模拟鱼群运动

1.4.2 人工生命研究进展

1.4.2.1 人工免疫系统（AIS）

人工免疫系统是一种遵循免疫学原理的自适应系统，广泛应用于不同的研究领域。它主要包括三个核心：克隆选择、阴性选择和免疫网络。克隆选择是抗体形成理论。阴性选择是免疫细胞排除重复字段、得到与目前所有抗原决定簇都不匹配的新受体的算法。免疫网络则是对克隆选择的补充，免疫过程中，抗体不仅显示出抗体活性，还显示出抗原活性，从而激发一系列免疫连锁反应。

基于以上原理构建的人工免疫识别系统模型，学习速度快，分类准确率高，可以较好地模拟真实免疫过程，并可以预测记忆细胞的存储过程。

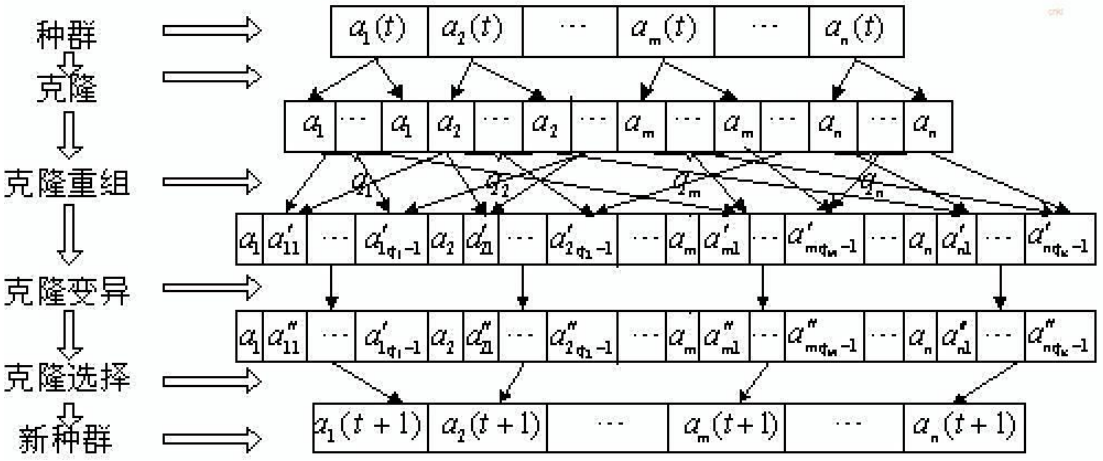


图 1-5 人工免疫系统算法的主要操作过程（刘若辰，2005）

1.4.2.2 神经网络结构设计

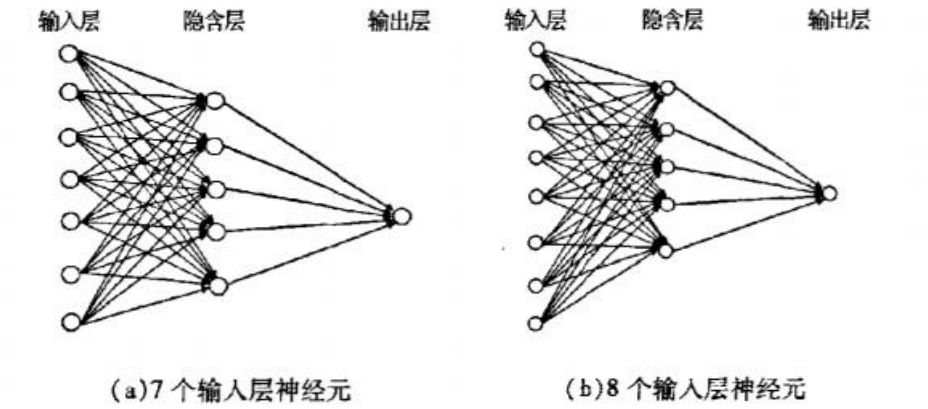


图 1-6 包含不同输入层神经元的网络结构（包芳，2009）

神经网络结构采用的比较广泛的是进化算法，结合细胞自动机构建交叉模型，可以实现更稳定、效率更高的查找策略。在神经网络中，神经元独立工作，每个神经元在特定模拟次数输出的信息仅与连接到它的神经元输入的有效信息有关。细胞自动机的邻域演化规则与此特性相当契合，适合用来模拟和优化神经元的信息传递过程。

神经网络结构设计可以较简洁地模拟神经网络输入、输出信息的过程，对于人工智能的自我学习和训练有重要的影响。不断优化的算法也让人工智能的更新迭代的效率越来越高。

1.4.3 人工生命发展前景

人工免疫系统、神经网络结构设计等方法和研究偏于计算，其关键在于构建合适的运算模型和算法。通过计算机高速而强大的计算能力和可操作的模拟，可以有效直观地验证生命活动。

至于其未来的发展，一方面，基于计算机硬件计算能力的提升，人工生命会集合更多生物相关的参数，更加拟真，预测更细致的生命活动。另一方面，优化的算法可以让模拟的效率更高、结果更准确和具有代表性。而且，基于虚拟现实等技术的发展，人工生命会在可视化方面进一步发展。

二、材料与方法

2.1 细胞自动机

本研究的基本思路是，抽取细胞集落为细胞自动机的细胞空间，构建考虑细胞的二维空间排列和细胞间相互作用的模型，通过使用随机过程和动态系统从群落的角度预测细胞相互作用。单个细胞由空间网格中的格子表示，每个细胞具有其私有属性，如寿命、年龄和对周围细胞的影响程度，以及其邻居，可以是细胞、屏障或细胞外基质。为了保持细胞生物学方面的真实性，在此程序模型中引入的规则和参数与细胞自动机的通常规则和参数有所区别。

分裂参数

参数名	参数意义
cycle	细胞分裂的阈值，指细胞周期。
age	浮点数，记录细胞的年龄，随模拟次数增长。 当 Age> Cycle 时，会进入可增殖状态。
growth	Age 在模拟次数方面的增长率，仅取决于所观察的细胞。
incidence	浮点数，衡量相邻细胞对 Age 生长速率的影响程度。 取决于相邻的细胞。
reaction	浮点数，衡量细胞对相邻细胞增殖率的反应程度。 取决于所观察的细胞和相邻细胞。
migration	整数,衡量观察细胞的迁移能力。记录细胞周围是否有空位。当细胞进入可增殖状态时，其周围必须有空间安置分裂的新细胞，才会进行分裂。（如果没有足够空间，则必须等到空余空间出现才分裂。）

死亡参数

参数名	参数意义
vitality	浮点数，记录细胞的活力，随模拟次数下降。当 vitality<0，观察的细胞会凋亡。
reduction	vitality 随模拟次数的下降率，仅取决于所观察细胞。
incidence_d	浮点数，衡量相邻细胞对 vitality 下降率的影响程度。取决于相邻的细胞。
reaction_d	浮点数，衡量细胞对相邻细胞凋亡率的影响程度。取决于观察到的细胞和相邻细胞。

细胞间相互作用的抑制和促进关系参数

参数名	参数意义
relation	描述细胞间相互对 age 作用的关系是抑制还是促进, 0 促进，1 抑制。
relation_d	描述细胞间相互对 vitality 作用的关系是抑制还是促进, 0 促进，1 抑制。

初等细胞自动机、生命游戏等模型只分析细胞邻居数量对细胞生命的影响，并没有明确引入年龄、活力等参数。本模型则从细胞自身出发，每个细胞都有明确的细胞周期、年龄、活力、增长速率、凋亡速率等属性。而且本模型引入了细胞相互作用关系的参数，能够更仿真地模拟细胞增殖的情景以及不同种类细胞之间的相互作用。

具体规则如下：

每个格子具有坐标[x][y]，代表其位置，以及当前状态（0，1，2），0 表示该格子没有细胞存活，1 和 2 表示当前该位置的细胞种类是 cell_1 还是 cell_2。

当细胞年龄随模拟次数增长，成为成熟具有分裂能力的细胞时，判断其周围是否有空的位置，只有相邻位置有状态为空的空间才能分裂。当细胞年龄过大、活力过低就会凋亡。age 随模拟次数增加，vitality 随模拟次数减少，这两个参数都会受到周围细胞的影响，如果 age 先达到 cycle 就准备增殖，如果 vitality 到 0 就凋亡。

细胞间的相互影响采用线性模拟。基本公式如下：

$$age = \sum_0^t growth;$$

$$vitality = V0 - \sum_0^t reduction;$$

$$incidence = avg\ (reaction);$$

$$incidence\_d = avg\ (reaction\_d);$$

当 relation = 0 时，growth += incidence;

当 relation = 1 时，growth -= incidence;

当 relation_d = 0 时，reduction += incidence_d;

当 relation_d = 1 时，reduction -= incidence_d.

本研究是基于模拟次数的模拟，不代入实际的时间参数。V0 是细胞初始的活力值。t 是模拟的次数。incidence 是相邻细胞的 reaction 的平均数，促进和抑制的影响分别计算。incidence_d 是相邻细胞 reaction_d 的平均数。cycle、age、

vitality 是和模拟次数 t 同级的参数；growth、incidence、reaction、reduction、incidence_d、reaction_d 是和 t 的倒数同级的参数；migration、relation、relation_d 是状态参数，只区分 0 或 1。

2.2 实验方法

实验程序是基于面向对象算法和 python 构建的，细胞静态类图见图 2-1。

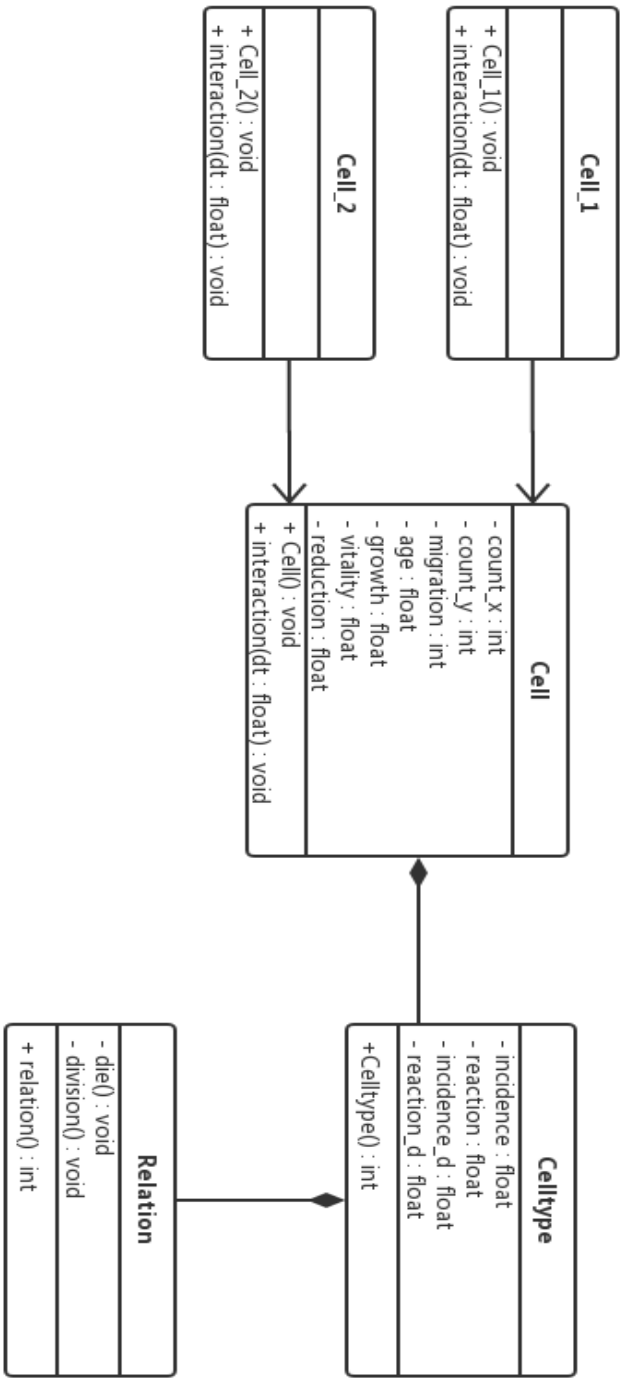


图 2-1 细胞自动机程序类图

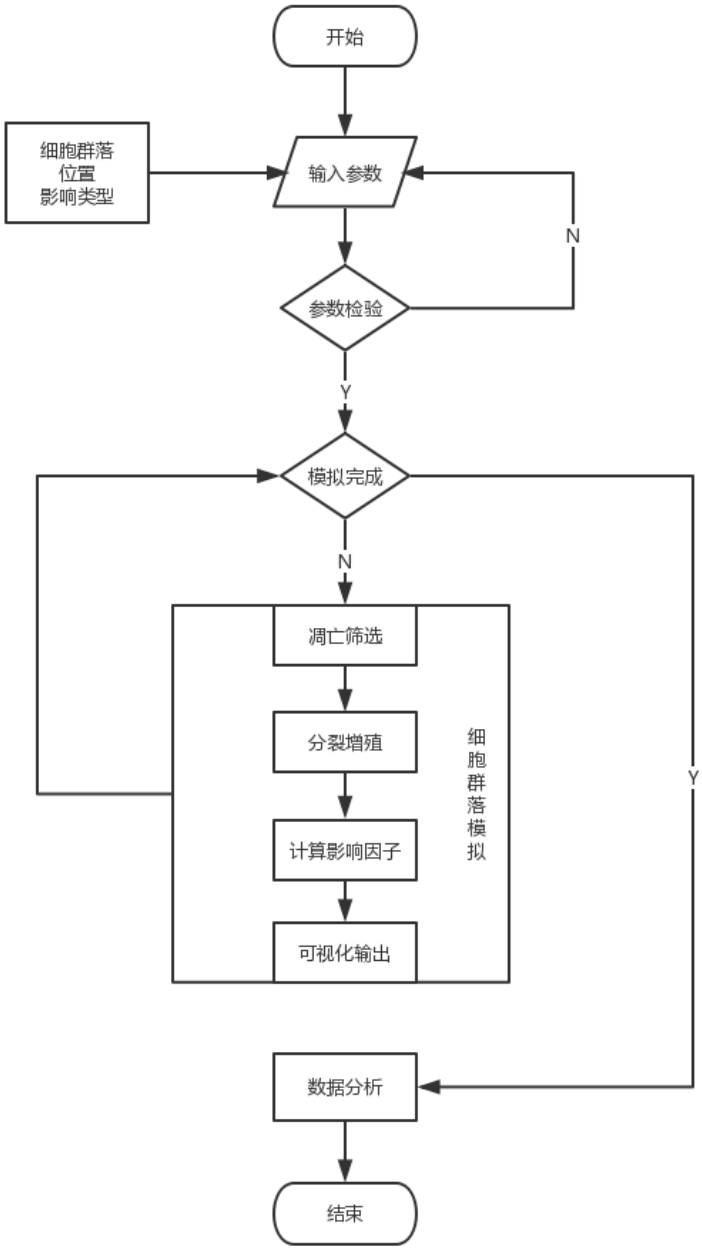


图 2-2 细胞群体行为模拟过程。

首先初始化系统。输入初始细胞参数（位置、寿命、类型等），设定其相互作用类型，就可以开始模拟迭代。每次分裂之后，程序会通过函数来记录每个细胞的一般状态，包括指数、位置、年龄和死亡或分裂倾向。模拟结束后，可以分析和验证程序记录的数据。实验流程如图 2-2。

模拟过程之后，观察记录图像可以粗略得出其增殖趋势，包括细胞集落的位置信息和生长方向。还需要进行细胞集落的定量分析，统计细胞存活数量，分析其数量变化趋势。

最后通过所得数据以及分析的结果，可以验证预设的细胞群体行为规律。

三、研究结果

3.1 程序模拟干细胞增殖

参数：cycle = 2; growth = 1.0; vitality = 9.0; reduction = 1.0。

自定义细胞位置：[16][16] = 1

其余参数为零。模拟细胞周期为 2、生长速率 1.0、初始活力 9.0、凋亡速率 1.0 的细胞。

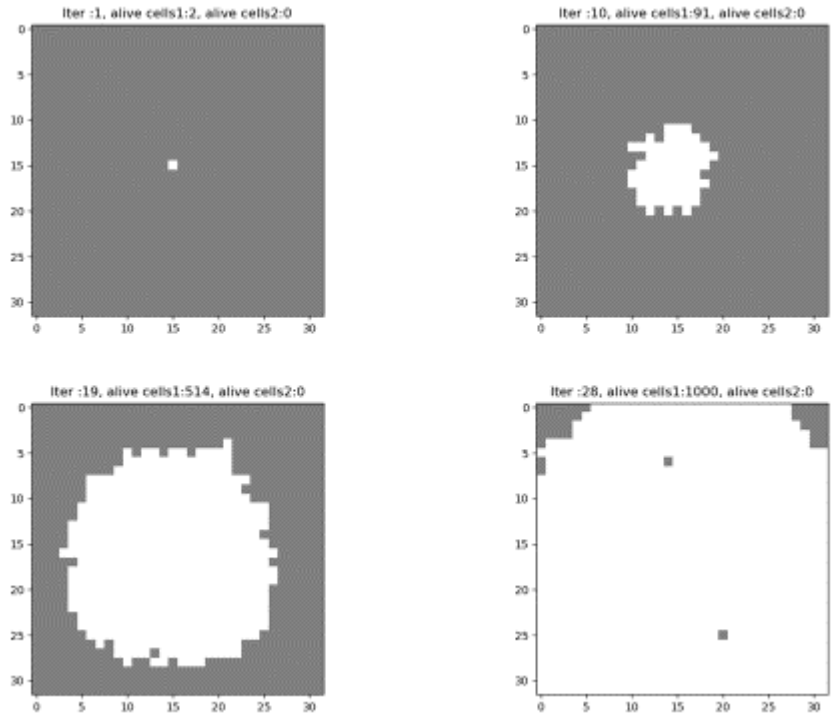
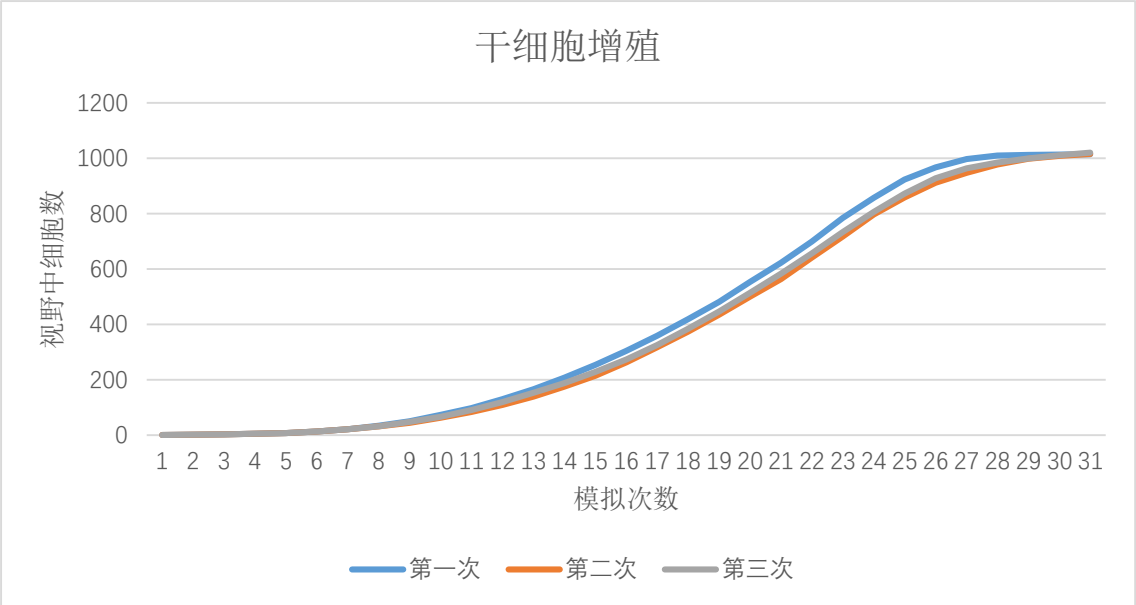


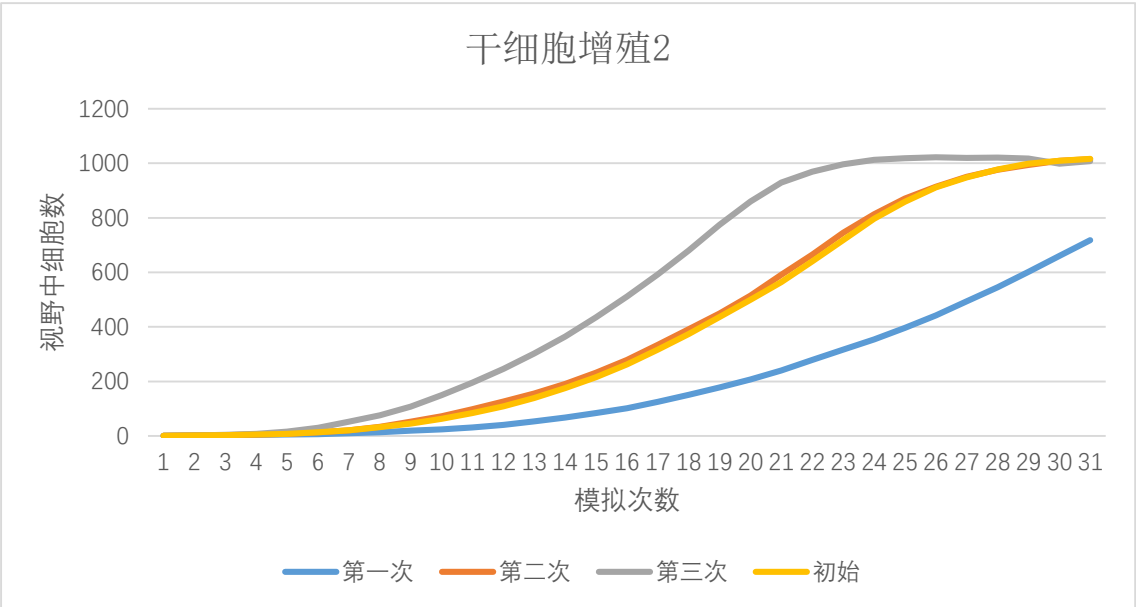
图 3-1 程序中的增殖演示，模拟次数 t 分别为 1，10，19，28
白色格子为干细胞，灰色为培养基背景。每张图标注了模拟次数和存活的细胞数量



保持 vitality、reduction 不变，改变 cycle 和 growth 的值。

	cycle	growth
第一次	4	1
第二次	4	2
第三次	2	2

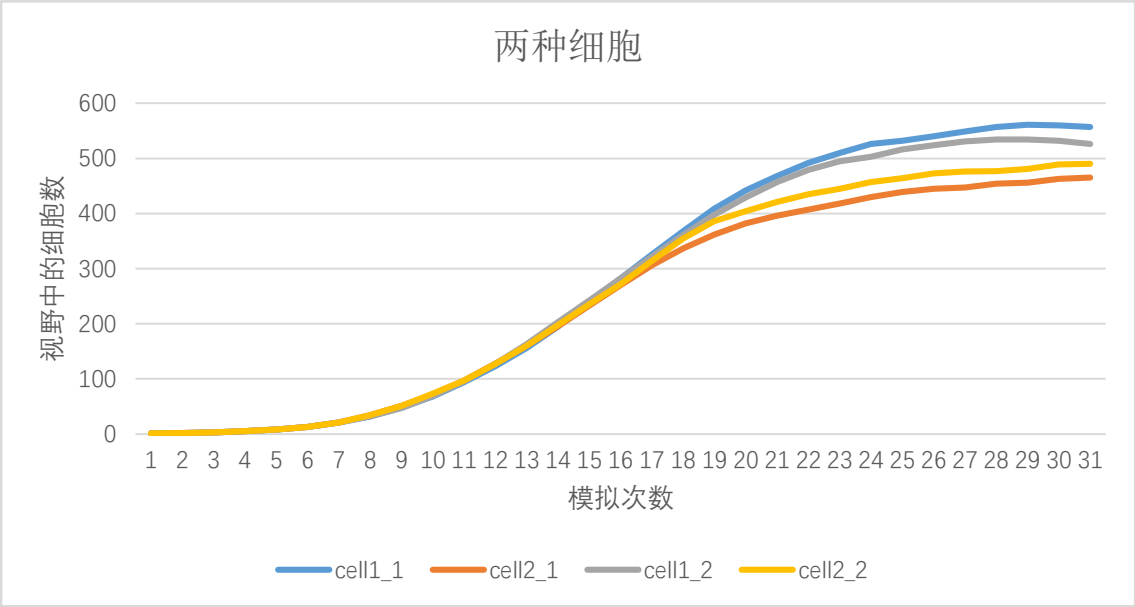
经过多次验证，细胞增殖速率与 growth/cycle 的比值正相关，可以通过调整比值模拟增殖能力不同的细胞。



3.2 程序模拟两种细胞增殖

首先模拟两种增殖速率相同、互不影响的正常细胞。

cycle = 4; growth = 2.0; vitality = 9.0; reduction = 1.0。其余参数为零。



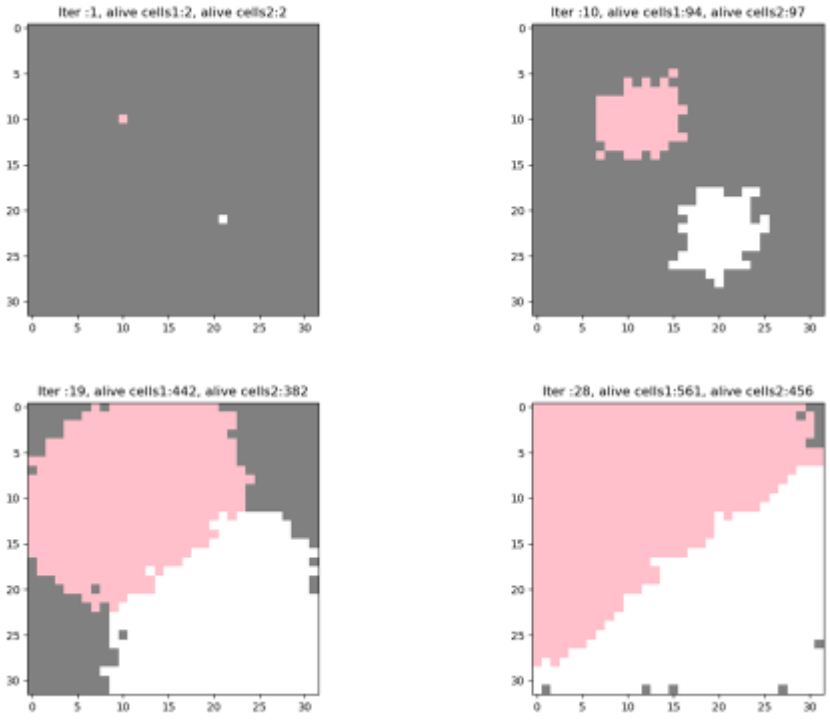


图 3-2 两种互不影响的细胞增殖模拟

红色为 cell1，白色为 cell2,模拟次数 t 分别为 1，10，19，28

发现两种互不影响的细胞增殖时相对平均，在有限空间内各占一半。通过调整 cycle 和 growth 的值以模拟增殖速率不同但不会相互影响的细胞。

	cycle	growth
cell1	4	2
cell2_1	3	2.5
cell2_2	1	1
cell2_3	1	1

当两种细胞较为靠近，会发现增殖速率快的细胞占极大优势。例如肿瘤细胞，细胞周期短，增殖速率快。cell1 模拟正常细胞，cell2 模拟肿瘤细胞。

前两次模拟发现，增加肿瘤细胞增殖速率，肿瘤细胞数量增加不明显。而后更改细胞初始位置，从视野左上、右下且间距较大，改成右上、左下且间距为之前的 1/4。（坐标由 [11][11] = 1；[22][22] = 2 变为 [14][16] = 1；[16][14] = 2）发现结果有更加显著变化，见图 3-3。

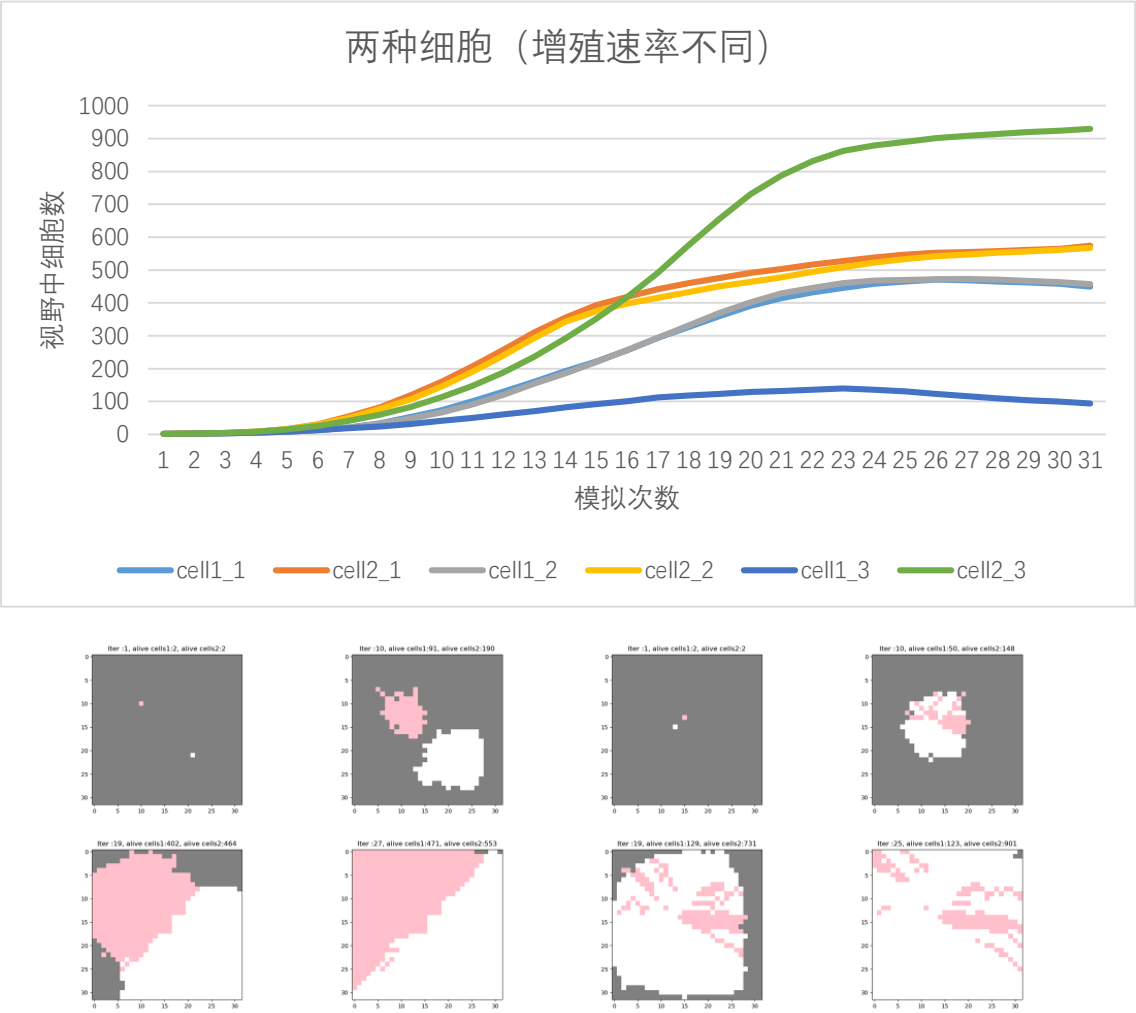


图 3-3 初始细胞放置的位置会对模拟结果产生影响
红色为 cell1，白色为 cell2,模拟次数 t 分别为 1，10，19，28

3.3 模拟两种细胞相互作用

根据单种及两种互不影响的细胞增殖模拟情况，设定适合增殖模拟的初始参数：

cycle=6　growth=2.0　vitality=9.0　reduction=1.0

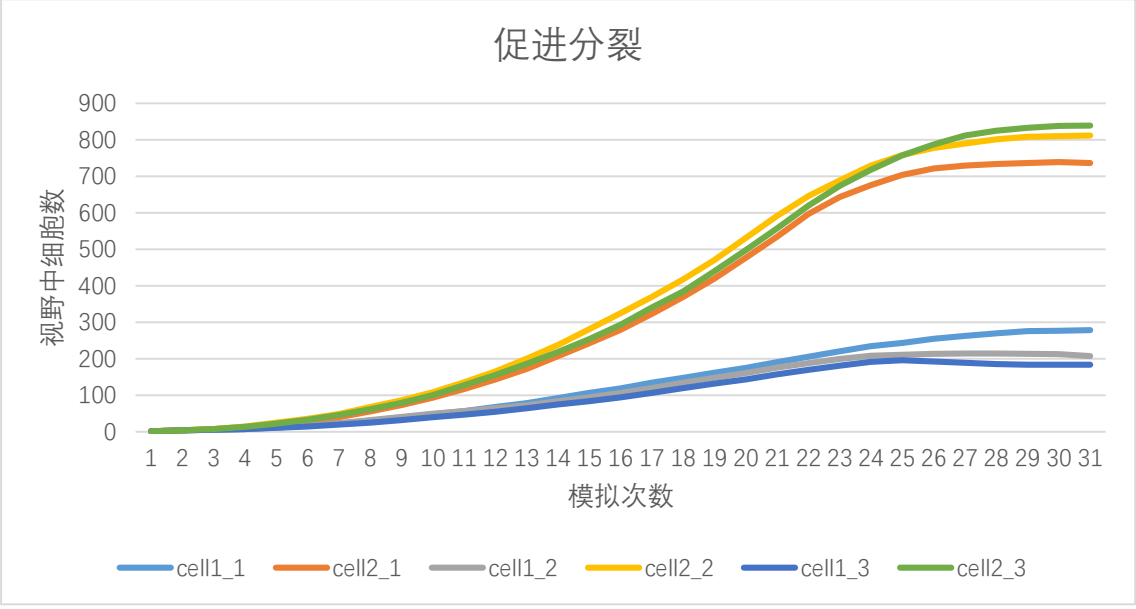
自定义初始化细胞位置，上半部分为 cell1，下半部分为 cell2：

[15][15] = 1　[15][16] = 1　[16][15] = 2　[16][16] = 2

需要调整的参数有：reaction，relation，reaction_d，relation_d。这些参数会影响细胞之间是促进还是抑制彼此的生长速率和凋亡速率，它们初始值都为 0。预设 cell1 为具有影响因子的细胞，cell2 是受到 cell1 的影响，增殖速率或凋亡速率相应改变的细胞。因为有参数 relation 判断是促进还是抑制细胞增殖，如果 relation=0，判断细胞会促进周围细胞分裂，而影响程度 reaction 小于 0，则实际结果为抑制细胞生长，可以放在 relation=1 即抑制分裂的情况里分析，参数 reaction_d 同理。故 reaction、reaction_d 只取正值。

3.3.1 促进分裂

调整 reaction 的值为 1，2，4。reaction 是衡量细胞对相邻细胞增殖率的反应程度的参数。其余参数保持为 0。



当 reaction 参数越大，cell_1 对 cell_2 增殖速率促进效果越明显。

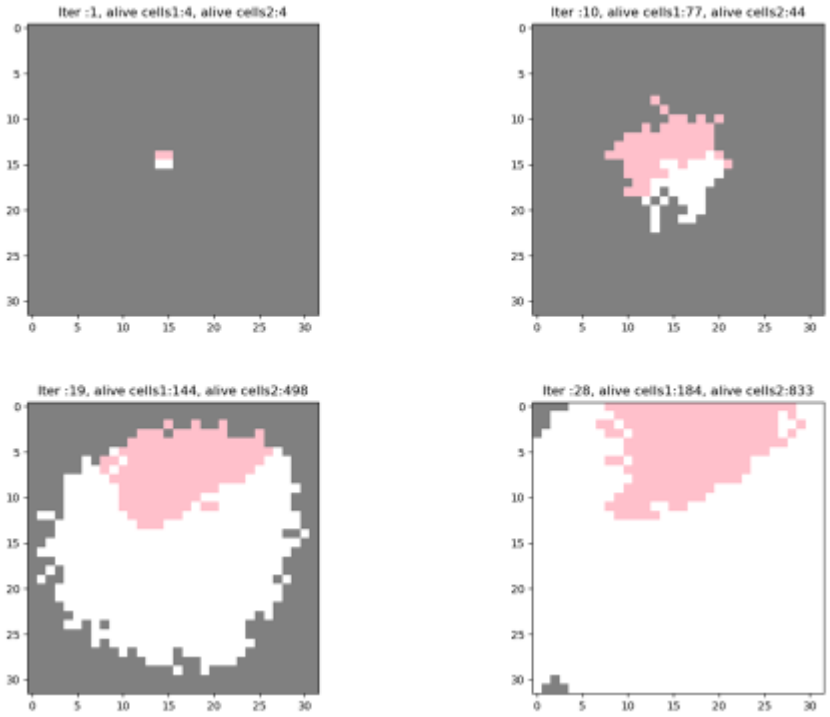
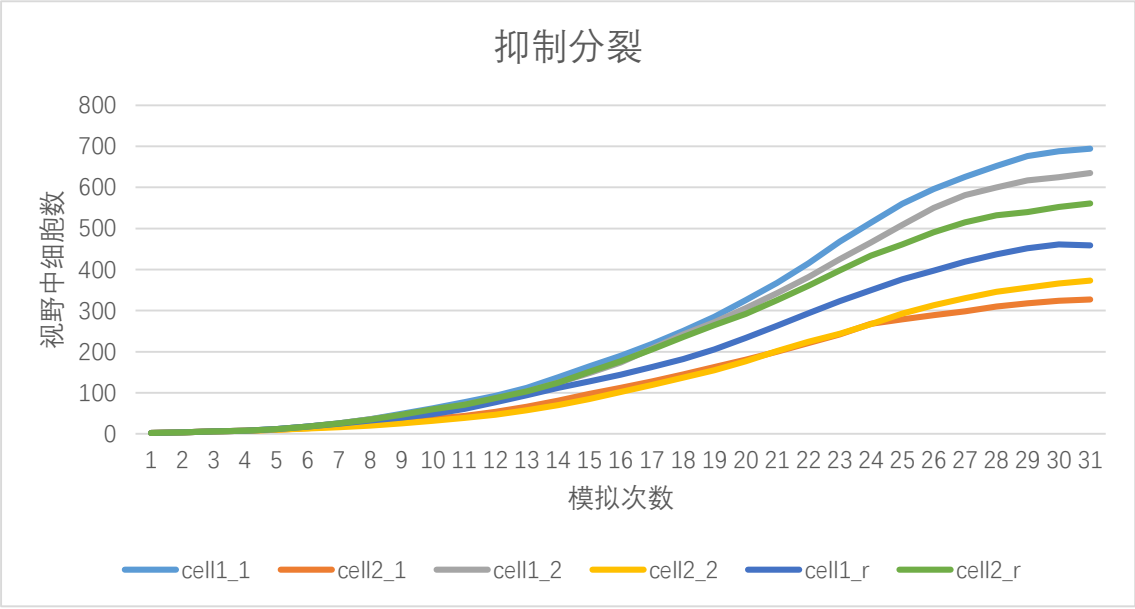


图 3-4 促进分裂，reaction=4,
红色为 cell1，白色为 cell2,模拟次数 t 分别为 1，10，19，28

3.3.2 抑制分裂

设置 relation=1（抑制增长速率）；reaction=1，1.5。



cell1_r、cell2_r 是不设置参数的对照组，即两种增殖速率相同且不相互影响的细胞的模拟。对比发现，当 reaction 为 1 时，cell_1 对 cell_2 的增殖抑制效果

最明显。

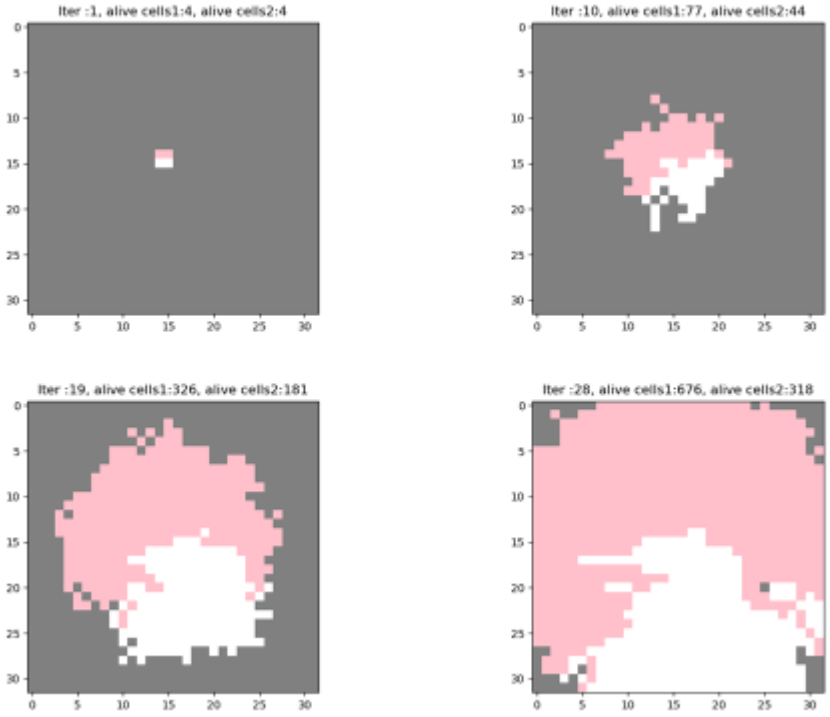
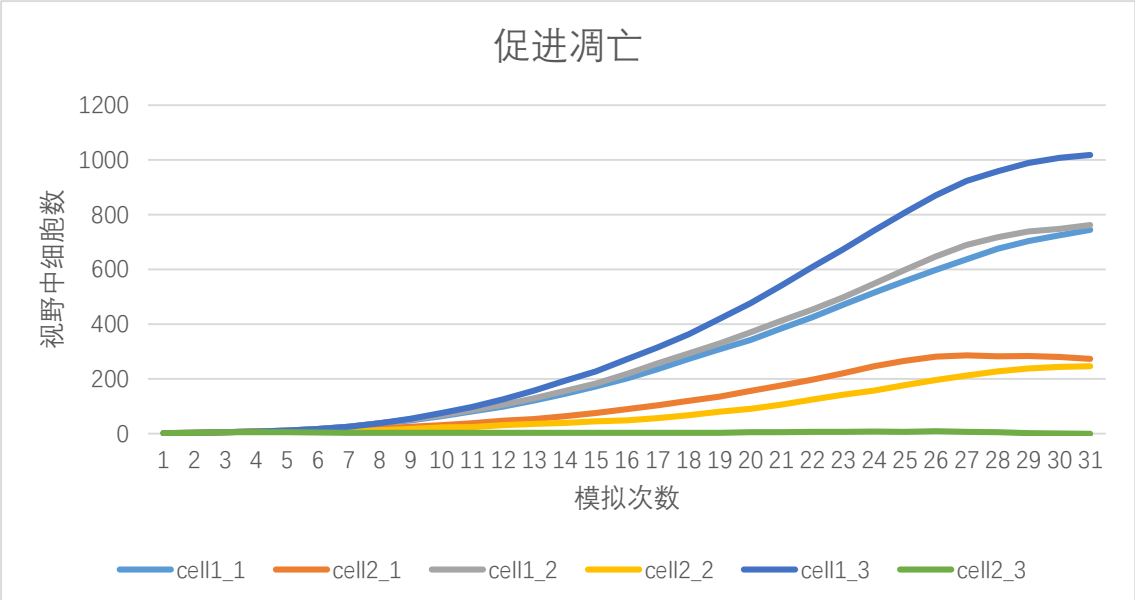


图 3-5 抑制分裂，reaction=1

红色为 cell1，白色为 cell2,模拟次数 t 分别为 1，10，19，28

3.3.3 促进凋亡

调整 reaction_d 的值为 0.5，1，2。reaction_d 是衡量细胞对相邻细胞凋亡速率的反应程度的参数。其余参数保持为 0。



cell_2 受到 cell_1 促进凋亡的影响，当 reaction_d 为 2 时，cell_2 几乎无法增殖，凋亡促进效果明显。

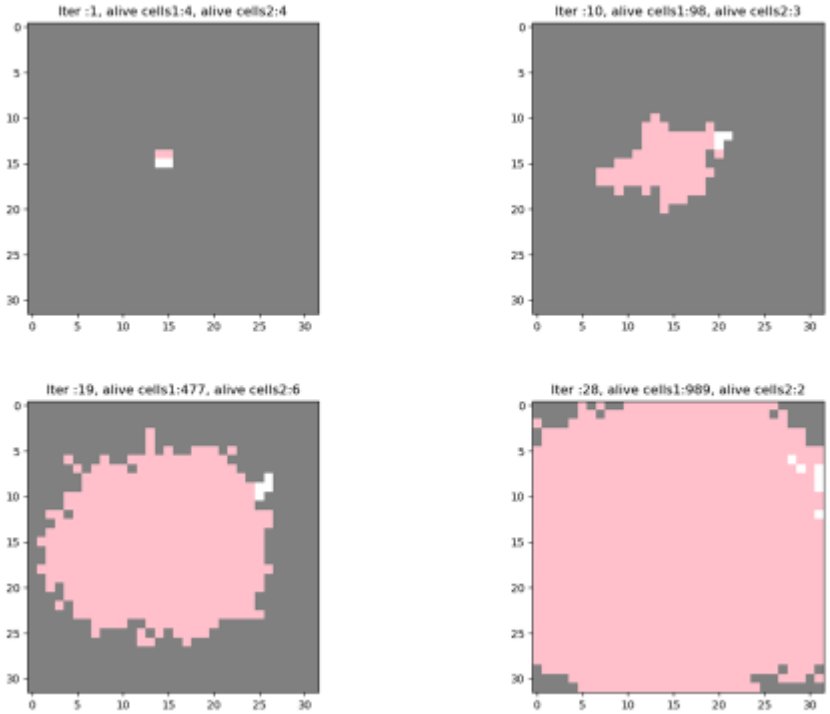
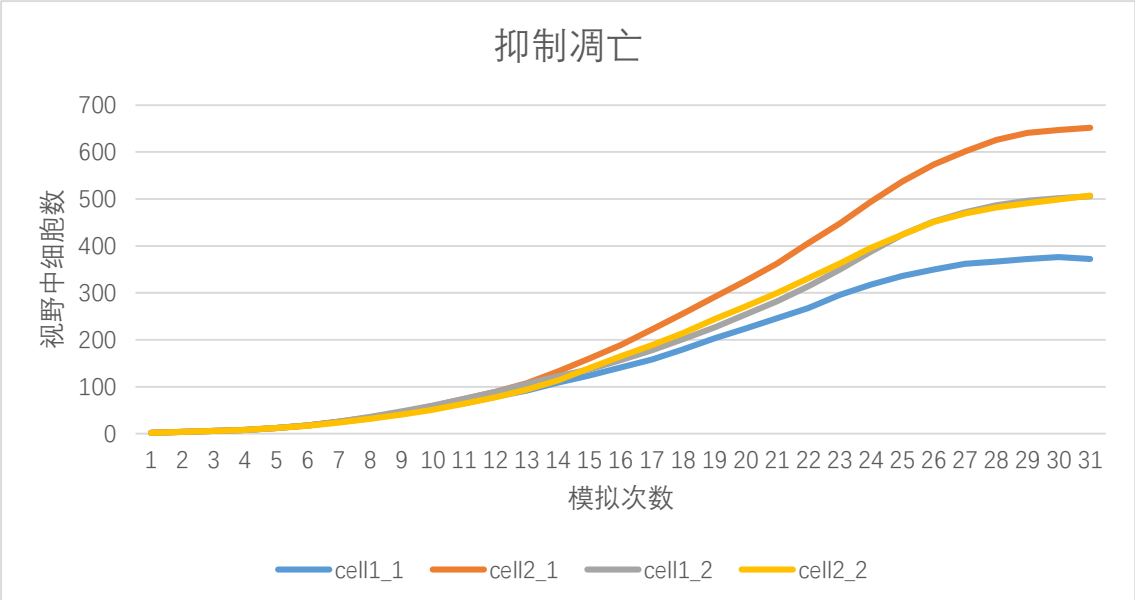


图 3-6 促进凋亡，reaction=2
红色为 cell1，白色为 cell2,模拟次数 t 分别为 1，10，19，28

3.3.4 抑制凋亡

relation_d = 1（抑制凋亡速率）；调整 reaction_d 的值为 0.5，0.8。



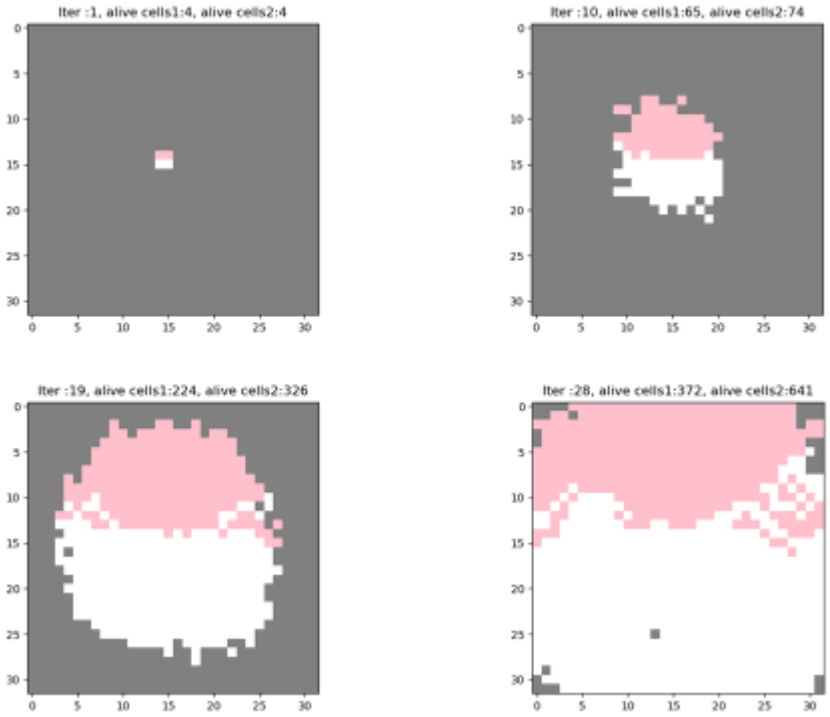


图 3-6 抑制凋亡，reaction=0.5

红色为 cell1，白色为 cell2,模拟次数 t 分别为 1，10，19，28

发现当 reaction_d=0.5 时，cell2 数量有显著增长。reaction_d 参数过大反而失去影响。

四、讨 论

4.1 微观规律分析

在微观层面，细胞群体行为受细胞信息交换和环境因素的影响。细胞能够通过信息分子感知环境，通过分泌、接收各种各样的化合物行使不同类型的相互作用。这种相互作用可以相互促进、对抗或中立。本研究通过程序成功模拟了细胞的各种群体行为，验证了细胞生命活动的基本规律。

细胞初始位置对其后续增殖有一定的影响。在干细胞增殖模拟中，新生细胞需要一定的模拟次数生长成熟之后才具有分裂增殖能力。当细胞周围布满细胞、没有剩余空间时，就不会继续分裂，符合接触抑制原则。如果细胞可以分裂，但是周围一直没有剩余空间，就会在活力（vitality）归零后凋亡。

分裂过程，具有分裂能力的细胞会在其周围随机挑选一个空位置生成新细胞。新的细胞从 0 开始，原位置的细胞继续生长。通过改变细胞周期（cycle）的值进行一系列测试，可以验证，细胞增殖会受到细胞周期的影响。

cycle、growth、vitality 的比值关系：growth 与 cycle 的比值体现了细胞增殖速率，呈正相关。vitality 与 cycle 的比值可以反映细胞分裂次数的上限，age 和 cycle 的比值则是细胞当前分裂的次数。

模拟普通细胞与肿瘤细胞竞争时，肿瘤细胞相比正常细胞，细胞周期短（cycle 值更小，growth 值更大），繁殖速度快，同时活力保持模拟次数长，在竞争中占优势。这是肿瘤细胞迅速增殖并持久活跃的主要原因。

模拟细胞间相互促进和抑制的相互作用，本模型提供了两个方面的思路。促进增殖，可以提高 growth，加快细胞成长速度；也可以降低 reduction，减缓细胞凋亡过程。抑制细胞增殖则相反。

4.2 宏观规律分析

生态学的宏观规律为研究细胞群体行为提供了新的思路。将细胞群体行为和生物种间关系对比，细胞群体行为所验证的规律与生态学种群规律存在关联性。

观察细胞数量曲线，很容易联想到生态学中种群增长的 J 曲线和 S 曲线。起初因为空间、营养等资源充足，细胞增殖符合指数规律，主要受到细胞总数的限

制；之后由于细胞总量增多，细胞间竞争加剧，增长率趋于平缓。竞争关系在群落之间和在细胞集落之间都普遍存在。其它的种间关系，如协作、捕食等，也能在细胞群体行为中找到相关联的关系模式。

种群间的信息交流与细胞群体行为中的信息交流也有共通之处。在演化群落竞争和交流的模型中，群落的规模取决于群落间的竞争，种群变异的压力来自其生存空间，种群间的信息交换通过种群的个体交流实现。而在细胞群体行为过程中，细胞集落的规模同样取决于细胞间的竞争，细胞个体间的信息交换组成了细胞集落间的信息交流，体现出促进或者抑制的关系。

种群的演化是种群间相互作用和自然选择两个层面的动态过程，细胞增殖也不仅是个体细胞自身的动力学过程，还是不同细胞集落间、细胞集落与环境之间相互作用和交流的结果。

4.3 改进方法

本研究在计算机上实现了基于二进制的细胞群体行为的模拟，该程序也有很大的优化空间。

首先，可以考虑优化细胞相互作用的规则。目前该程序是较为简单的线性拟合，如果采用更复杂的拟合公式，会有更好的拟合结果。该程序是较为理想状态下对细胞群体行为的模拟，没有考虑温度等环境因素对细胞增殖的影响。而且因为模拟的细胞总分裂次数较少，不需要考虑正常细胞癌变的情况。

关于边界情况，本模型将边界视为培养基的内壁，细胞无法穿透内壁。模拟过程中，四个角的细胞只有三个邻居，四条边上的细胞只有五个邻居，处于中间的细胞才有八个邻居。可以模拟无边界的情况，设置边界循环情况，连接对应的边和角，以保证每个细胞都有八个能相互影响的邻居。

程序方面，可以进一步优化程序图形界面。同时，本程序留出了对应接口，可以拓展成三种及以上细胞相互作用的模拟。

4.4 结论

本研究将软件工程与生物工程相结合，在细胞自动机的基础上，研究更符合真实的细胞活动情景。本研究的重心在于细胞相互作用，通过确定细胞自身属性

以及相互作用关系，程序可以自动模拟细胞增殖过程并输出结果。

本研究通过计算机程序模拟了细胞自我复制、细胞之间竞争、细胞之间相互促进和抑制增殖等细胞群体行为。微观上符合实验所得的细胞增殖和相互作用规律。从整体的角度观察多细胞相互作用，程序也能够在宏观尺度得到相对正确的模拟，并且可以用生态学理论将细胞集落和种群的种间关系联系起来。

细胞群体行为，既是基于细胞自身的动力学过程，也是符合宏观生态学规律的群体行为。本研究是基于细胞自动机和细胞生物学原理的拓展，在技术和操作层面有较高的可行性。程序具有较好的可视化效果，具有较好的可重复性和可操作性，并且具有很大的进步空间。

参考文献

[1] 刘天庆,葛丹,程昉,鞠智好,李香琴,孙相,马学虎,崔占峰. 神经干细胞球生长动力学模型[J]. 中国生物医学工程学报,2006,(06):708-716.
[2] 潘宏伟. 粘球菌及其细胞群体行为环境适应的分子机制研究[D].山东大学,2010.
[3] 李广运,韩力群. 人工生命及其基本原理[J]. 计算机仿真,2004,(05):121-125.
[4] 杨琛璐. 基于二维细胞自动机的蚁群聚类研究及应用[D].西安电子科技大学,2013.
[5] 陈林,杨亮,段康民. 从进化谈细菌细胞间的群体感应信号传递[J]. 遗传,2012,34(01):35-42.
[6] 邓泽林,谭冠政,何锫. 基于记忆细胞剪切和非线性资源分配的人工免疫识别系统[J]. 模式识别与人工智能,2013,26(04):374-381.
[7] 包芳,潘永惠,孙俊,须文波. 基于细胞自动机和 QPSO 的间接编码神经网络结构设计算法[J]. 模式识别与人工智能,2009,22(01):148-155.
[8] 班晓娟,陈希,宁淑荣. 具有自身平衡系统的虚拟生物在三维空间内捕食与逃逸的关键技术研究[J]. 计算机科学,2009,36(09):234-237.
[9] 宋蓓,张国军,李芬芬,李虓,康熙雄. 全自动数字细胞形态学分析系统 Cella Vision~(TM) DM96 在外周血白细胞分类的临床应用评价[J]. 检验医学与临床,2015,12(04):481-483.
[10] 刘若辰. 免疫克隆策略算法及其应用研究[D].西安电子科技大学,2005.
[11] 周成虎.地理元胞自动机研究[M].科学出版社,1999.
[12] Sprinzak David,Lakhanpal Amit,Lebon Lauren,Santat Leah A,Fontes Michelle E,Anderson Graham A,Garcia-Ojalvo Jordi,Elowitz Michael B. Cis-interactions between Notch and Delta generate mutually exclusive signalling states.[J]. Nature,2010,465(7294):.

致谢

在此毕业论文完成之际，我突然回忆起大学四年的时光。生物科学方面的专业课程的教导，让我拥有严谨的实验科研态度，以及保持着对知识的渴求。

感谢蔡亮老师从立题开始对我认真的指导和督促。论文方向的确定源于我对程序方面的兴趣，我希望在生物和编程结合的方向做出一点探索。蔡亮老师给了我许多富有远见的建议，并给予了我大力的支持。

同时也感谢我的同学们给我的研究提出的意见和建议。感谢我的朋友们对我的支持。感谢金晏伊同学在我研究期间的陪伴和鼓励。

附录

实验代码

Cell.py

```
class Cell():
    """class cell
    cycle          细胞分裂的阈值，指细胞周期。
    age              浮点数，记录细胞的年龄，随模拟次数增长。 当 age>Cycle 时，可能会发生分裂。
    growth           age 在模拟次数方面的增长率，仅取决于观察到的细胞。
    incidence   浮点数，衡量相邻细胞对 age 生长速率的影响程度。 取决于相邻的细胞。
    reaction    浮点数，衡量细胞对相邻细胞增殖率的反应程度。 取决于观察到的细胞和相邻细胞。
    migration   整数,衡量观察细胞的迁移能力。记录细胞周围是否有空位，如果有才能分裂。migration 表示当 age> Cycle 时，观察到的细胞如何为分裂留出空间。（如果空间失败，则单元格必须等到可访问空间出现。）

    vitality      浮点数，记录细胞的年龄，随模拟次数下降。当 Vitality<0，观察的细胞会凋亡。
    reduction   vitality 随模拟次数的下降率，仅取决于观察到的细胞。
    incidence_d浮点数，衡量相邻细胞对 vitality 下降率的影响程度。取决于相邻的细胞。
    reaction_d  浮点数，衡量细胞对相邻细胞增值率的影响程度。取决于观察到的细胞和相邻细胞。
    相互作用
    relation      描述细胞间相互对 age 作用的关系是抑制还是促进,0 促进，1 抑制。
    relation_d   描述细胞间相互对 vitality 作用的关系是抑制还是促进,0 促进，1 抑制。
    """

    def __init__(self, cycle=6, iteraction=0, age=0.0, growth=2.0, incidence=0, reaction=0, migration=0,
vitality=9.0,
                 reduction=1.0, incidence_d=0, reaction_d=0, relation=0, islive=0, relation_d=0, type=1):
        """init a cell"""
        self.cycle = cycle
        self.age = age
        self.growth = growth
        self.incidence = incidence
        self.migration = migration
        self.vitality = vitality
        self.reaction = reaction
        self.reduction = reduction
        self.incidence_d = incidence_d
        self.reaction_d = reaction_d
        self.islive = islive
        self.iteraction = iteraction

        self.type = type
        if(type == 1):
            self.reaction = 0
            self.reaction_d = 0.8
            self.relation = 0
            self.relation_d = 0
        elif(type == 2):
            ##self.cycle = 1
            ##self.growth = 1.0
            ##self.vitality = 50.0
            self.reaction = 0
            self.reaction_d = 0
            self.relation = 0
            self.relation_d = 1
```

CellDivision.py

```
from Cell import Cell
import random
import numpy as np
import matplotlib.pyplot as plt
import matplotlib as mpl
```

```
class CellDivision(object):
    def ini(self):
        """自定义初始化细胞位置"""
        self.cells[15][16] = 1
        self.cells[16][15] = 2
        self.cells[15][15] = 1
        self.cells[16][16] = 2


    def rand(self):
        """随机初始化细胞位置"""
        self.cells[1:-1, 1:-1] = np.random.randint(2, size=(self.cells_width, self.cells_height))


    def __init__(self, cells_shape):
        self.cells = np.zeros(cells_shape)
        self.cells_width = cells_shape[0] - 2
        self.cells_height = cells_shape[1] - 2
        self.ini()    ##self.ini()
        self.timer = 0
        self.cellArray = []
        self.alive_cell_num = [0,0,0,0,0,0];
        for count_x in range(0, self.cells_width + 2):
            self.cellArray.append([]);
            for count_y in range(0, self.cells_height + 2):
                if (self.cells[count_x, count_y]):
                    self.cellArray[count_x].append(Cell(islive=1,type=self.cells[count_x, count_y]))
                    num = int(self.cells[count_x, count_y])
                    self.alive_cell_num[num] += 1
                else:
                    self.cellArray[count_x].append(Cell(islive=0, type=0))


    def die(self):
        """所有的细胞生长 1 个周期，不计算所有的影响，只剔除死亡的细胞"""
        for count_x in range(0, self.cells_width + 1):
            for count_y in range(0, self.cells_height + 1):
                cell = self.cellArray[count_x][count_y]
                if (cell.islive):
                    """只有活着才生长"""
                    cell.vitality -= cell.reduction
                    cell.age += cell.growth
                    if (cell.vitality < 0):
                        print("（{}，{}）死去".format(count_x, count_y))
                        self.cellArray[count_x][count_y].islive = -1
                        num = int(self.cellArray[count_x][count_y].type)
                        self.alive_cell_num[num] -= 1
                        self.cells[count_x][count_y] = 0
                    print(
                        "（{}，{}），age={}, incidence={}, vitality={}, incidence_d={},
type={}".format(count_x, count_y,

cell.age,

cell.incidence,

cell.vitality,

cell.incidence_d,

cell.type))
                    if ((cell.age / cell.cycle) > cell.iteraction) and (cell.vitality > 0):
                        self.cellArray[count_x][count_y].migration = 1
                else:
                    self.cells[count_x][count_y] = 0


    def division(self):
```

```
        """当细胞可以分裂，且周围有空余空间，分裂一个新的细胞,新的细胞从 0 开始，原位置的细胞继续生长"""
        for count_x in range(1, self.cells_width + 1):
            for count_y in range(1, self.cells_height + 1):
                cell = self.cellArray[count_x][count_y]
                if(cell.islive == -1):
                    cell.islive = 0
                elif self.cellArray[count_x][count_y].migration == 1:
                    tmp = 0
                    enable = int(self.cells[count_x][count_y])

                    if count_x == 1:
                        if count_y == 1:
                            if (self.cellArray[1][2].islive * self.cellArray[2][2].islive * self.cellArray[2][1].islive == 0):
                                while self.cellArray[count_x][count_y].migration == 1:
                                    i = random.randint(0,1) + 1
                                    j = random.randint(0,1) + 1
                                    if (self.cellArray[i][j].islive == 0):
                                        if (self.cellArray[i][j].islive == 0):
                                            print("x={},y={},type={}出生".format(i, j, enable))
                                            cell_n = Cell(islive=1,type=enable)
                                            self.cellArray[i][j] = cell_n
                                            self.cells[i][j] = enable
                                            self.alive_cell_num[enable] += 1
                                            self.cellArray[count_x][count_y].iteraction += 1
                                            self.cellArray[count_x][count_y].migration = 0

                        elif count_y == self.cells_height:
                            if (self.cellArray[1][self.cells_height-1].islive * self.cellArray[2][self.cells_height-1].islive * self.cellArray[2][self.cells_height].islive == 0):
                                while self.cellArray[count_x][count_y].migration == 1:
                                    i = random.randint(0,1) + 1
                                    j = random.randint(0,1) + self.cells_height - 1
                                    if (self.cellArray[i][j].islive == 0):
                                        print("x={},y={},type={}出生".format(i, j, enable))
                                        cell_n = Cell(islive=1,type=enable)
                                        self.cellArray[i][j] = cell_n
                                        self.cells[i][j] = enable
                                        self.alive_cell_num[enable] += 1
                                        self.cellArray[count_x][count_y].iteraction += 1
                                        self.cellArray[count_x][count_y].migration = 0

                        else:
                            for i in range(1, 3):
                                for j in range(count_y - 1, count_y + 2):
                                    if (self.cellArray[i][j].islive == 0):
                                        tmp += 1
                            if tmp :
                                while self.cellArray[count_x][count_y].migration == 1:
                                    i = random.randint(0,1) + 1
                                    j = random.randint(0,2) + count_y - 1
                                    if (self.cellArray[i][j].islive == 0):
                                        print("x={},y={},type={}出生".format(i, j, enable))
                                        cell_n = Cell(islive=1,type=enable)
                                        self.cellArray[i][j] = cell_n
                                        self.cells[i][j] = enable
                                        self.alive_cell_num[enable] += 1
                                        self.cellArray[count_x][count_y].iteraction += 1
                                        self.cellArray[count_x][count_y].migration = 0

                    elif count_x == self.cells_width:
                        if count_y == 1:
                            if (self.cellArray[self.cells_width][2].islive * self.cellArray[self.cells_width-1][2].islive * self.cellArray[self.cells_width-1][1].islive == 0):
                                while self.cellArray[count_x][count_y].migration == 1:
                                    i = random.randint(0,1) + self.cells_width -1
                                    j = random.randint(0,1) + 1
                                    if (self.cellArray[i][j].islive == 0):
```

```
                                        print("x={},y={},type={}出生".format(i, j, enable))
                                        cell_n = Cell(islive=1,type=enable)
                                        self.cellArray[i][j] = cell_n
                                        self.cells[i][j] = enable
                                        self.alive_cell_num[enable] += 1
                                        self.cellArray[count_x][count_y].iteraction += 1
                                        self.cellArray[count_x][count_y].migration = 0

                        elif count_y == self.cells_height:
                            if (self.cellArray[self.cells_width][self.cells_height-1].islive *
self.cellArray[self.cells_width-1][self.cells_height-1].islive * self.cellArray[self.cells_width-
1][self.cells_height].islive == 0):
                                while self.cellArray[count_x][count_y].migration == 1:
                                    i = random.randint(0,1) + self.cells_width - 1
                                    j = random.randint(0,1) + self.cells_height - 1
                                    if (self.cellArray[i][j].islive == 0):
                                        print("x={},y={},type={}出生".format(i, j, enable))
                                        cell_n = Cell(islive=1,type=enable)
                                        self.cellArray[i][j] = cell_n
                                        self.cells[i][j] = enable
                                        self.alive_cell_num[enable] += 1
                                        self.cellArray[count_x][count_y].iteraction += 1
                                        self.cellArray[count_x][count_y].migration = 0

                        else:
                            for i in range(count_x - 1, count_x + 1):
                                for j in range(count_y - 1, count_y + 2):
                                    if (self.cellArray[i][j].islive == 0):
                                        tmp += 1
                            if tmp :
                                while self.cellArray[count_x][count_y].migration == 1:
                                    i = random.randint(0,1) + count_x - 1
                                    j = random.randint(0,2) + count_y - 1
                                    if (self.cellArray[i][j].islive == 0):
                                        print("x={},y={},type={}出生".format(i, j, enable))
                                        cell_n = Cell(islive=1,type=enable)
                                        self.cellArray[i][j] = cell_n
                                        self.cells[i][j] = enable
                                        self.alive_cell_num[enable] += 1
                                        self.cellArray[count_x][count_y].iteraction += 1
                                        self.cellArray[count_x][count_y].migration = 0

                    elif count_y == 1:
                        for i in range(count_x - 1, count_x + 2):
                            for j in range(1, 3):
                                if (self.cellArray[i][j].islive == 0):
                                        tmp += 1
                            if tmp :
                                while self.cellArray[count_x][count_y].migration == 1:
                                    i = random.randint(0,2) + count_x - 1
                                    j = random.randint(0,1) + 1
                                    if (self.cellArray[i][j].islive == 0):
                                        print("x={},y={},type={}出生".format(i, j, enable))
                                        cell_n = Cell(islive=1,type=enable)
                                        self.cellArray[i][j] = cell_n
                                        self.cells[i][j] = enable
                                        self.alive_cell_num[enable] += 1
                                        self.cellArray[count_x][count_y].iteraction += 1
                                        self.cellArray[count_x][count_y].migration = 0

                    elif count_y == self.cells_height:
                        for i in range(count_x - 1, count_x + 2):
                            for j in range(count_y - 1, count_y + 1):
                                if (self.cellArray[i][j].islive == 0):
                                    tmp += 1
                            if tmp :
                                while self.cellArray[count_x][count_y].migration == 1:
                                    i = random.randint(0,2) + count_x - 1
                                    j = random.randint(0,1) + count_y - 1
```

```
                                        if (self.cellArray[i][j].islive == 0):
                                            print("x={},y={},type={}出生".format(i, j, enable))
                                            cell_n = Cell(islive=1,type=enable)
                                            self.cellArray[i][j] = cell_n
                                            self.cells[i][j] = enable
                                            self.alive_cell_num[enable] += 1
                                            self.cellArray[count_x][count_y].iteraction += 1
                                            self.cellArray[count_x][count_y].migration = 0

                        else:
                            for i in range(count_x - 1, count_x + 2):
                                for j in range(count_y - 1, count_y + 2):
                                    if (self.cellArray[i][j].islive == 0):
                                        tmp += 1
                            if tmp :
                                while self.cellArray[count_x][count_y].migration == 1:
                                    i = random.randint(0,2) + count_x - 1
                                    j = random.randint(0,2) + count_y - 1
                                    if (self.cellArray[i][j].islive == 0):
                                        print("x={},y={},type={}出生".format(i, j, enable))
                                        cell_n = Cell(islive=1,type=enable)
                                        self.cellArray[i][j] = cell_n
                                        self.cells[i][j] = enable
                                        self.alive_cell_num[enable] += 1
                                        self.cellArray[count_x][count_y].iteraction += 1
                                        self.cellArray[count_x][count_y].migration = 0


    def caculate(self, count_x, count_y):
        sum_1 = 0;
        sum_2 = 0;
        sum_d_1 = 0;
        sum_d_2 = 0;
        incidence_1 = 0;
        incidence_2 = 0
        incidence_d_1 = 0;
        incidence_d_2 = 0;
        for i in range(count_x - 1, count_x + 2):
            for j in range(count_y - 1, count_y + 2):
                if (self.cells[i][j] and (i != count_x or j != count_y) and (self.cellArray[i][j].type !=
self.cellArray[count_x][count_y].type)):
                    if (self.cellArray[count_x][count_y].relation == 0):
                        sum_1 += 1
                        incidence_1 += self.cellArray[i][j].reaction
                    elif (self.cellArray[count_x][count_y].relation):
                        sum_2 += 1
                        incidence_2 += self.cellArray[i][j].reaction
                    if (self.cellArray[count_x][count_y].relation_d == 0):
                        sum_d_1 += 1
                        incidence_d_1 += self.cellArray[i][j].reaction_d
                    elif (self.cellArray[count_x][count_y].relation_d):
                        sum_d_2 += 1
                        incidence_d_2 += self.cellArray[i][j].reaction_d
        if (sum_1):
            self.cellArray[count_x][count_y].growth += incidence_1/sum_1
        if (sum_2):
            self.cellArray[count_x][count_y].growth -= incidence_2/sum_2
        if (sum_d_1):
            self.cellArray[count_x][count_y].reduction += incidence_d_1/sum_d_1
        if (sum_d_2):
            self.cellArray[count_x][count_y].reduction -= incidence_d_2/sum_d_2


    def effect(self):
        """计算相邻细胞的影响因素"""
        for count_x in range(1, self.cells_width + 1):
            for count_y in range(1, self.cells_height + 1):
                cell = self.cellArray[count_x][count_y]
                if (self.cells[count_x, count_y]):
```

```
                        self.caculate(count_x=count_x, count_y=count_y)


    def update_state(self):
        self.die()
        self.division()
        self.effect()
        self.timer += 1


    def update_and_plot(self, n_iter):

        plt.ion()
        self.effect()
        colorslist = ['gray', 'pink', '#ffffff']
        cmaps = mpl.colors.LinearSegmentedColormap.from_list('mylist', colorslist, N=800)
        for _ in range(n_iter):
            plt.title('Iter :{}, alive cells1:{}, alive cells2:{}'.format(self.timer, self.alive_cell_num[1],
self.alive_cell_num[2]))
            buf = self.cells[1:self.cells_width + 1, 1:self.cells_height + 1]
            plt.savefig(str(format(self.timer))+'.png')
            plt.imshow(buf, cmap=cmaps)
            self.update_state()
            plt.pause(0.2)
            with open('14-1.txt','a') as f:
                f.write("{}\n".format(self.alive_cell_num[1]))
            with open('14-2.txt','a') as g:
                g.write("{}\n".format(self.alive_cell_num[2]))
            print("----------{}----------".format(self.timer))
            print("alive cells1:{}, alive cells2:{}".format(self.alive_cell_num[1], self.alive_cell_num[2]))
        plt.ioff()



if __name__ == '__main__':
    cell = CellDivision(cells_shape=(34, 34));
    cell.update_and_plot(30)
##print(Cell().__dict__);
```