使用细胞自动机模拟简单生态系统

冯国蕴　　林欣煜　　马詠汛　　谢金宏

2019 年 12 月 21 日

摘要

在本次课题中，我们小组使用 Python 语言实现了基本的细胞自动机。并在已有的细胞自动机的规则基础上进行扩展，尝试对具有氧气、生产者和消费者三要素的生态系统进行模拟，并将模拟的结果以图像形式表示。

目录

插图

1 细胞自动机与康威生命游戏

细胞自动机[1]最早由冯·诺依曼在 1950 年代为模拟生物细胞的自我复制而提出，起初未受到科学界的广泛关注。后因约翰·何顿·康威设计了生命游戏[2]而闻名于世。本节将简要介绍细胞自动机与康威生命游戏。

1.1 细胞自动机

我们可以在二维的格状棋盘上实现细胞自动机。在这个二维的空间上，棋盘中每个格子内细胞的状态是有限的，细胞下一时刻的状态由该细胞的邻居在当前的时刻的状态决定。棋盘内的所有细胞遵守相同的演化规则。

一个细胞自动机一般具有以下特点：

- **平行计算** 每个细胞个体的状态都同步地进行改变。
- **局部性** 细胞的状态只受相邻的细胞的影响。
- **一致性** 所有的细胞受到相同的规则约束。

1.2 康威生命游戏

康威生命游戏符合细胞自动机的特点，其规则定义如下：

- 每种细胞有“存活”和“死亡”两种状态。
- 当细胞周围的存活细胞数量等于 3 个时，细胞变为存活状态。
- 当细胞周围的存活细胞数量少于等于 1 个或大于等于 4 个时，细胞变为死亡状态。（模拟细胞过于孤独或环境过于拥挤。）
- 其他情况下细胞状态不变。

我们小组使用 Python 实现了经典的康威生命游戏（1.basic.py）。程序先随机地给定棋盘内细胞初始状态，然后按照康威生命游戏规则演化指定的代数，给出最终的结果。

图1是随机产生初始状态后按康威规则演化一代的结果。图中黑色块表示活细胞，白色块表示空位或死细胞。

1.3 镜像边界

细胞自动机理论上具有无穷大的“棋盘”作为细胞演化的空间，但计算机的内存空间是有限的，故无限大的棋盘不能实现，实际运行中的棋盘往往是规模为 n 的矩形棋盘。有限的空间就存在边界问题，边界上的细胞因此需要特殊处理。

在1.basic.py 中，边界上的细胞不参与演化，它们总是保持死细胞的状态。这是处理边界问题的一种方法，即边界上的细胞取得定值，在演化过程中保持不变。

另一种处理边界细胞问题的方法是在1.basic-mirror-edge.py 中实现的镜像边界。如图2所示，编号为 1 的细胞，它的左上角邻居、左邻居和右邻居分别被镜像地设置为 9 号、3 号和 7 号细胞。

[1]Cellular automaton https://en.wikipedia.org/wiki/Cellular_automaton

[2]Conway's Game of Life https://en.wikipedia.org/wiki/Conway's_Game_of_Life

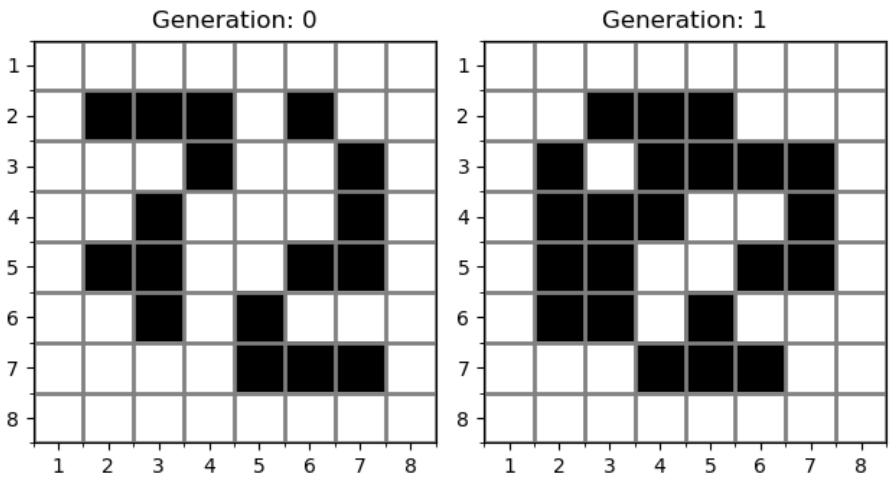

图 1: 按康威生命游戏规则进行一次迭代

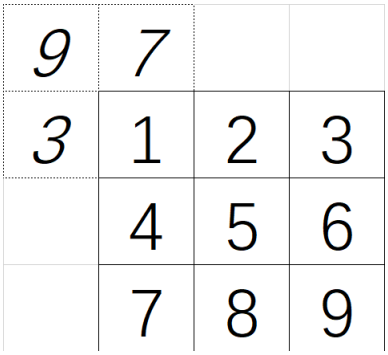

图 2: 镜像边界的一个实例

一般地，对于采用镜像边界的 $SIZE$ 大小的棋盘，格点位置为 (i, j) 的细胞，它的邻居可以通过如下程序遍历：

```
for i, j in CELLS:
  for dx in range(-1, 2):
    for dy in range(-1, 2):
      neighbor = board.iat[(i + dx + SIZE) % SIZE, (j + dy + SIZE) % SIZE]
```

如无特殊说明，本文后续的模型均采用镜像边界的处理方法。

1.4 康威生命游戏的几个经典图案

我们在程序中实现了康威生命游戏的几个经典图形。包括一些永远保持“静止”不发生变化的图像（如图3）和一些在有限的图形中顺序切换，周而复始的图形（如图4）。

2 氧气模型

氧气是模拟生态系统中的要素之一，本节将介绍我们小组定义的氧气模型。

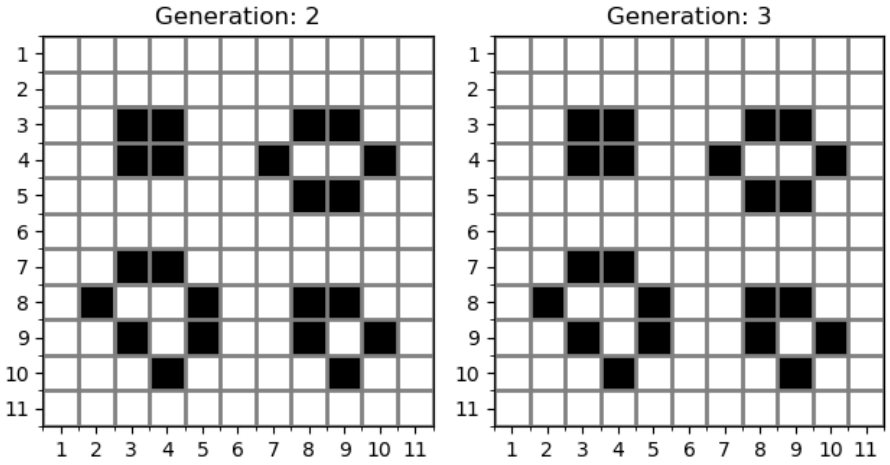

图 3: 永远保持静止的图形

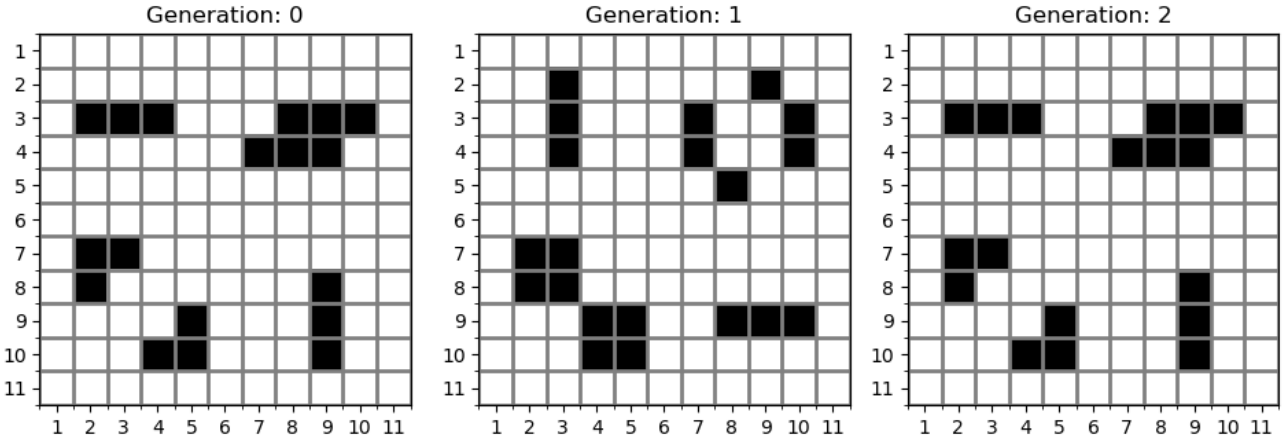

图 4: 周而复始的图形

2.1 扩散规则

在现实生活中，氧气可以在自由空间内扩散，最终平均扩散到空间各处，各处的氧气浓度一致。因此要模拟氧气，最重要的是模拟氧气随着时间的推移而扩散到自由空间内的性质。

我们使用迭代次数的增加表示时间的推移。棋盘上每个格子的体积一定，因此一个格子上所含有的氧气的量可以用氧气浓度的数值进行表示和参与运算。规定格子上氧气的浓度可以在某次迭代中超过 1，但不能在连续的几次迭代中持续超过 1。

对于一个格子而言，格子在下一轮氧气的浓度为上一轮氧气浓度减去本轮扩散到其他格子的氧气浓度，再加上本轮其他格子扩散到当前格子的氧气浓度。

规定一个格子 c 在一轮扩散中氧气的总扩散量为上轮氧气浓度 $last_c$ 的一半，对于与其相邻的包括它自身在内的 9 个格子 $neighbor(c)$，每个格子 i 从 c 中获取的氧气的扩散量 $shared_{c,i}$ 由以下公式进行计算：

$$shared_{c,i} = \frac{1}{2} \cdot last_c \cdot \frac{1 - min(1, last_i)}{\sum(1 - min(1, last_i))}, i \in neighbor(c) \tag{1}$$

上述公式的物理意义是，在氧气扩散总量一定的前提下，格子的原有浓度越小，所分得的氧气浓度就越高。公式中分子和分母位置的 $1 - min(1, last_i)$ 的含义是，氧气不向氧气浓度大于等于 1 的格子扩散。

那么对于一个格子 c，其本轮的氧气浓度由以下公式进行计算：

$$current_c = \frac{1}{2} \cdot last_c + shared_{i,c}, i \in neighbor(c) \tag{2}$$

按上面的公式1和公式2进行的计算是遵循物质守恒定律的。

对于棋盘边界上的格子，如果采用定值方法处理边界，则氧气浓度应该设置为 1，且边界格子不参与氧气扩散量的计算。由公式1知，边界格子亦不会吸收氧气。

如果采用镜像边界，则边界格子与非边界格子使用同样的规则进行迭代。

2.2 氧气模型的实现

我们在程序2.oxygen.py 中实现了上一小节中描述的氧气模型。

如图5所示的热力图中，蓝色表示氧气。格子中的氧气密度越高，格子上蓝色的程度就越深。如图，棋盘的初始状态随机给出，随后按照我们定义的氧气规则进行演化。

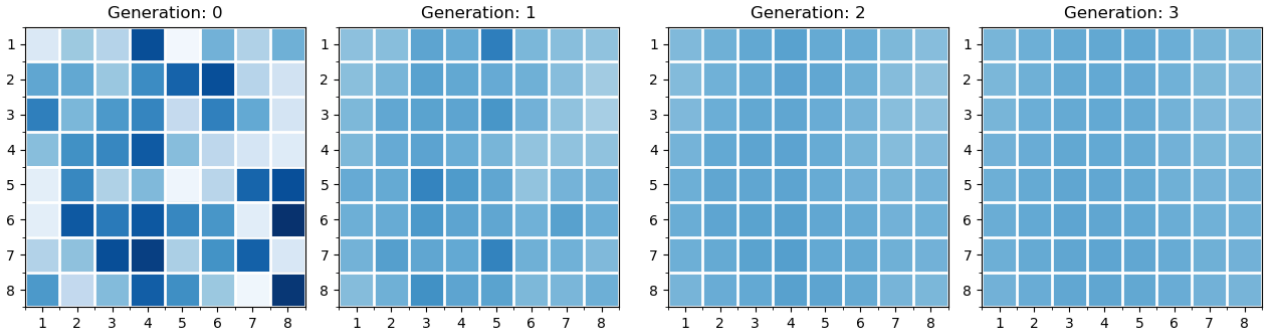

图 5: 氧气在自由空间内的扩散

在图6和图7中，我们还模拟了棋盘中氧气从一处或几处浓度峰点向外自由扩散的情况。

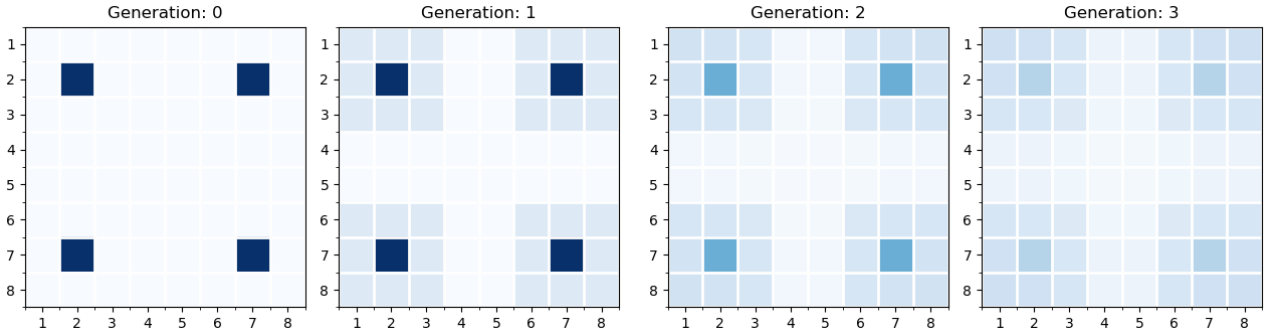

图 6: 氧气从峰值点的扩散 1

从图像上看，我们的氧气模型一定程度上成功地模拟了现实世界的情况。

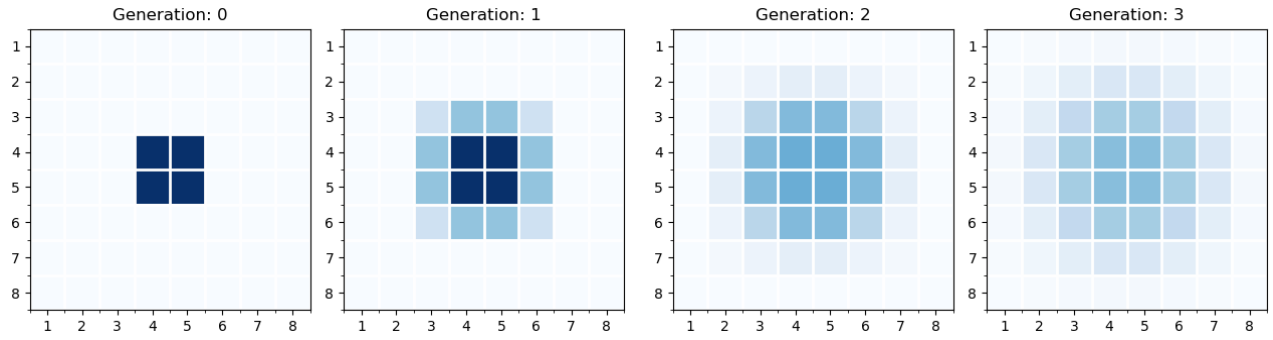

图 7: 氧气从峰值点的扩散 2

3 生产者模型

生态系统可分为生物部分和非生物部分。在上一节中，我们在非生物部分中选择了氧气这样要素进行了模拟；本节中，我们将模拟生物部分的基础——生产者。

3.1 氧气生产规则

生产者的基本功能是生产氧气。考虑生产者生产的氧气时，我们忽略生产者的呼吸作用，只考虑净光合作用的影响。

以下是光合作用的公式：

$$12H_2O + 6CO_2 \overset{sun}{\rightarrow} C_6H_{12}O_6 + 6O_2 + 6H_2O \tag{3}$$

从光合作用的公式来看，光合作用的原料是水和二氧化碳，条件是阳光，最终产物是葡萄糖、水和氧气。我们模拟的生态系统中，缺少对水和二氧化碳的模拟。而我们将光照强度 $sunLevel$ 作为影响生产者生产氧气速率的因素，其取值范围在 $[0,1]$ 上。光合作用的最终产物之一葡萄糖的贡献体现在下一小节中生产者种群密度的增长上。

与氧气浓度类似，我们定义单位格子上生产者的数量为种群密度 $producerDensity$，其取值范围在 $[0,1]$ 上。生产者的种群密度也是其生产氧气速率的影响因素之一。定义种群密度的临界值为 0.01，若低于 0.01，则认为当前格子上的生产者种群密度过低，不足以继续发展而消亡。

我们用迭代表示时间的推移，定义一个格子上的生产者每次迭代产生的氧气 $producedOxygen$（在数值上表现为氧气浓度的增量）按以下公式计算：

$$producedOxygen = sunLevel \cdot producerDensity \cdot max(1 - oxygenLevel, 0) \tag{4}$$

公式中 $oxygenLevel$ 为生产者所在格子的氧气浓度。公式的生物学含义是，在其他条件一定的前提下，氧气浓度越高（意味着二氧化碳的浓度越低），生产者产生氧气的能力就越受到限制。此公式也保证了生产者生产氧气后，其所在格子的氧气浓度不会因此而超过 1。

图8为程序3.producer.no-growth.py 模拟的生产者生产氧气的过程。程序先给出随机氧气分布然后开始迭代，氧气从生产者处产生，尔后扩散到环境中的其他部分。

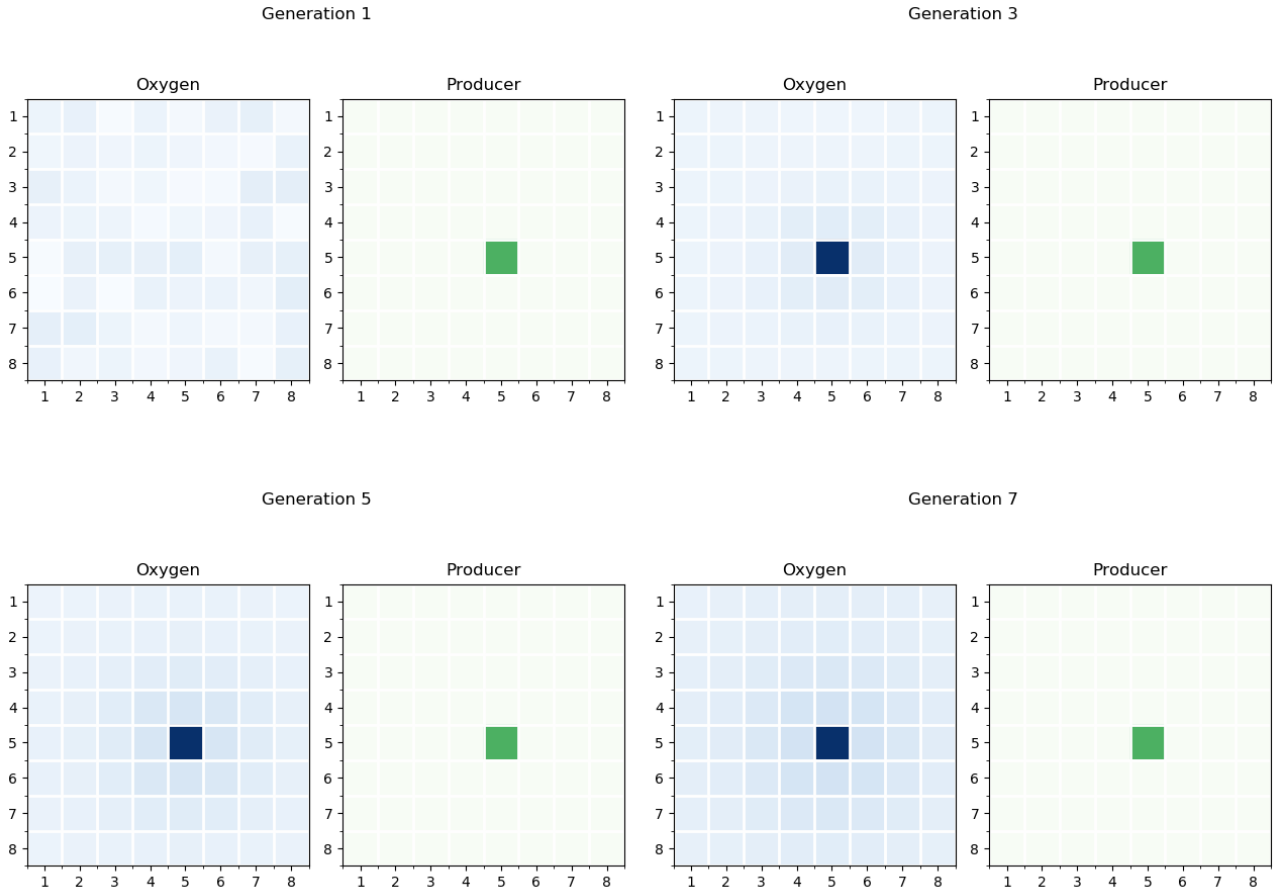

图 8: 生产者生产氧气

3.2 种群密度的增长规则

从生物学的角度上看，生物种群密度的增长随时间的推移而呈现出 S 型曲线，即种群的增长率一直下降，增长速率先增加后下降。

我们定义**理想种群密度曲线** $dt(x)$ 为二次曲线 $y = -(x-1)^2 + 1$。如图9所示，横轴为上次迭代中生产者的种群密度，纵轴为本次迭代生产者的种群密度。整条曲线在直线 $y = x$ 上方，表示种群密度保持增长；且曲线斜率不断下降，表示种群增长率不断下降。

$$dt := density_{current} = -(density_{last} - 1)^2 + 1 \tag{5}$$

为了体现光照强度对生产者生产的影响，在一次迭代中，生产者的实际种群密度使用下面的公式进行修正：

$$density_{current} = density_{last} + (dt(density_{last}) - density_{last}) \cdot sunLevel \tag{6}$$

3.3 繁衍规则

类似于康威生命游戏中细胞的增殖，我们模拟生态系统中的生产者按以下规则在新的格子上产生种群：设当前格子生产者的种群密度为 $density$，则在其不包括自身的 8 个邻居中，对于每个邻居格子，若

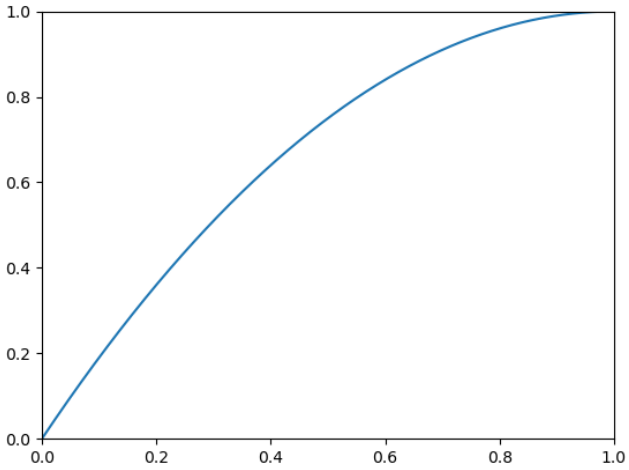

图 9: 理想种群密度曲线

此邻居格子上没有生产者，则有 $\frac{density}{8}$ 的概率在这个格子上产生种群密度为 0.02 的生产者。

按上述规则进行迭代，当前格子上生产者的种群密度越高，就越容易将种子传播到邻居格子处，播种出新的群落，这是符合常识的。

程序3.producer.py 模拟了生产者增长和繁衍的过程，其结果如图10所示。

4 消费者模型

消费者是生态系统的生物部分的另一个重要的角色；本节将介绍我们小组设计的消费者模型。

4.1 呼吸、捕食和种群增长

消费者不能自己进行光合作用，需要消耗氧气进行呼吸和食用生产者作为能量来源。呼吸作用是光合作用（公式3）的逆过程，从光合作用的公式来看，单位消费者需要消耗 0.5 单位的氧气和 0.5 单位的生产者来保证自身种群的增长。当氧气和生产者能够满足消费者的需求时，消费者也按照3.2小节中的种群密度曲线进行增长。定义消费者呼吸、捕食和种群增长的规则如下：

设当前格子消费者的种群密度为 $density$，则当前格子上的消费者需氧量 $needOxygen$（以氧气浓度计算）为 $\frac{1}{2} \cdot density$，需要的食物量 $needFood$（以生产者种群密度计算）为 $\frac{1}{2} \cdot density$。

1. 计算当前格子上当前种群密度的消费者的需氧量 $need$。若当前格子上实际含氧量 $actual$ 小于 $need$，则消费者的种群密度下降为 $density \cdot \frac{need-actual}{need}$，并消耗 $actual$ 量的氧气；否则消耗 $need$ 量的氧气。

2. 计算当前格子上当前种群密度的消费者的食物需求量 $need$（若第一步计算中消费者的种群密度下降了，则使用下降后的种群密度计算 $need$）。若当前格子上的生产者种群密度 $actual$ 小于 $need$，则生产者的种群密度下降为 $density \cdot (\frac{1}{2} + \frac{1}{2} \cdot \frac{need-actual}{need})$，并消耗 $actual$ 量的生产者；否则消耗 $need$ 量的生产者。

3. 若经过前两步计算后，消费者消耗了 $need$ 量的氧气和生产者，则表示环境适宜，消费者按照理想种群密度曲线进行增长（公式5）。

在我们设计的消费者规则中，氧气浓度比食物更重要些，缺乏氧气比缺乏食物更难存活。这是考虑到日常生活的经验中，生物缺少氧气会导致窒息死亡，但在食物短缺的情况下往往还能依据自身储备的碳量存活一段时间。

4.2 移动规则

消费者显著区别于生产者之处为，消费者可以进行移动，以寻找氧气和食物而不是坐以待毙。

消费者在移动时，将自己的 8 个邻居格子中本轮迭代和上轮迭代中没有其他消费者占据的邻居格子加入候选列表。若候选列表为空，则本轮不进行移动。若候选列表不为空，则按公式7逐个计算候选列表中格子的分数，然后选择得分最高的格子移动。

$$score = oxygenLevel \cdot weight_{oxygen} + producerLevel \cdot weightproducer \tag{7}$$

公式7中，$oxygenLevel$ 和 $producerLevel$ 分别为对应格子的氧气浓度和生产者种群密度。$weight$ 为权重。我们小组的程序中，氧气权重和生产者种群密度权重分别取值 0.6 和 0.4。

程序4.consumer.no-emerge.py 模拟了消费者的呼吸、捕食和种群增长和移动过程，其结果如图11所示。

4.3 繁衍规则

消费者也类似于生产者，具有繁衍种群的特性。设当前格子消费者的种群密度为 $density$，则在其不包括自身的 8 个邻居中，对于每个邻居格子，若此邻居格子上没有消费者，则有 $\frac{density}{8}$ 的概率在这个格子上产生种群密度为 0.02 的消费者。

程序4.consumer.py 模拟了完整的消费者模型，模拟结果如图12所示。

5 模拟生态系统

通过建立氧气模型、生产者模型和消费者模型，我们已经取得了模拟一个简单的生态系统所需要的全部积木。将这三片积木拼装在到一个棋盘中（或者说是在 z 轴方向层叠的三块棋盘上），即可模拟出一个简单的生态系统。为了平衡氧气的产生速率与生产者、消费者的活动速率，程序中计算迭代的顺序为：氧气迭代 4 次，生产者和消费者各迭代 1 次。在程序5.ecosystem.py 模拟了完整的生态系统。

利用该程序，并控制光照强度变量，我们获得了在三种不同强度的光照条件下模拟生态系统迭代 128 次后的图形，如图13、图14和图15所示。从图中可以很明显地看出不同光照条件下生态系统的差异。生态系统的光照条件越充足，所能承载的消费者种群密度就越高。

图16为一个模拟生态系统的演化过程。

6 结语

我们小组提出的关于生态系统的计算模型，从一定程度上模拟了现实世界中的生态系统。

我们也认识到提出的模型中存在有待商榷之处。例如，在基础的氧气模型中，我们规定单位格点的氧气密度不能超过 1，我们也尝试建立数学规则将氧气密度约束在 $[0, 1]$ 上。但在实际的实验数据中，总会有些意料之外的情形。这说明我们提出的氧气模型是存在一定的缺陷的。

我们在对细胞自动机进行“拓展”的时，其实已经破坏了细胞自动机的一些性质。举一例说，消费者的移动规则破坏了细胞自动机的平行计算性。因为在计算移动时，不仅使用了上一轮的数据，还使用了本轮的数据，因此遍历元胞的先后顺序会影响计算的结果。综上所述，或许不能称我们得到的模型为“细胞自动机”。

这篇报告所体现的内容并不是我们最初想法的全部。报告原题为《使用细胞自动机和遗传算法模拟简单生态系统》，原本设想是在建立氧气、生产者和消费者模型的基础上，结合遗传算法，模拟在不同的环境参数下，自然选择和生物进化的过程。但编程过程中出现了一些问题。时间不足，只好作罢。希望日后有机会继续完善这个模型。

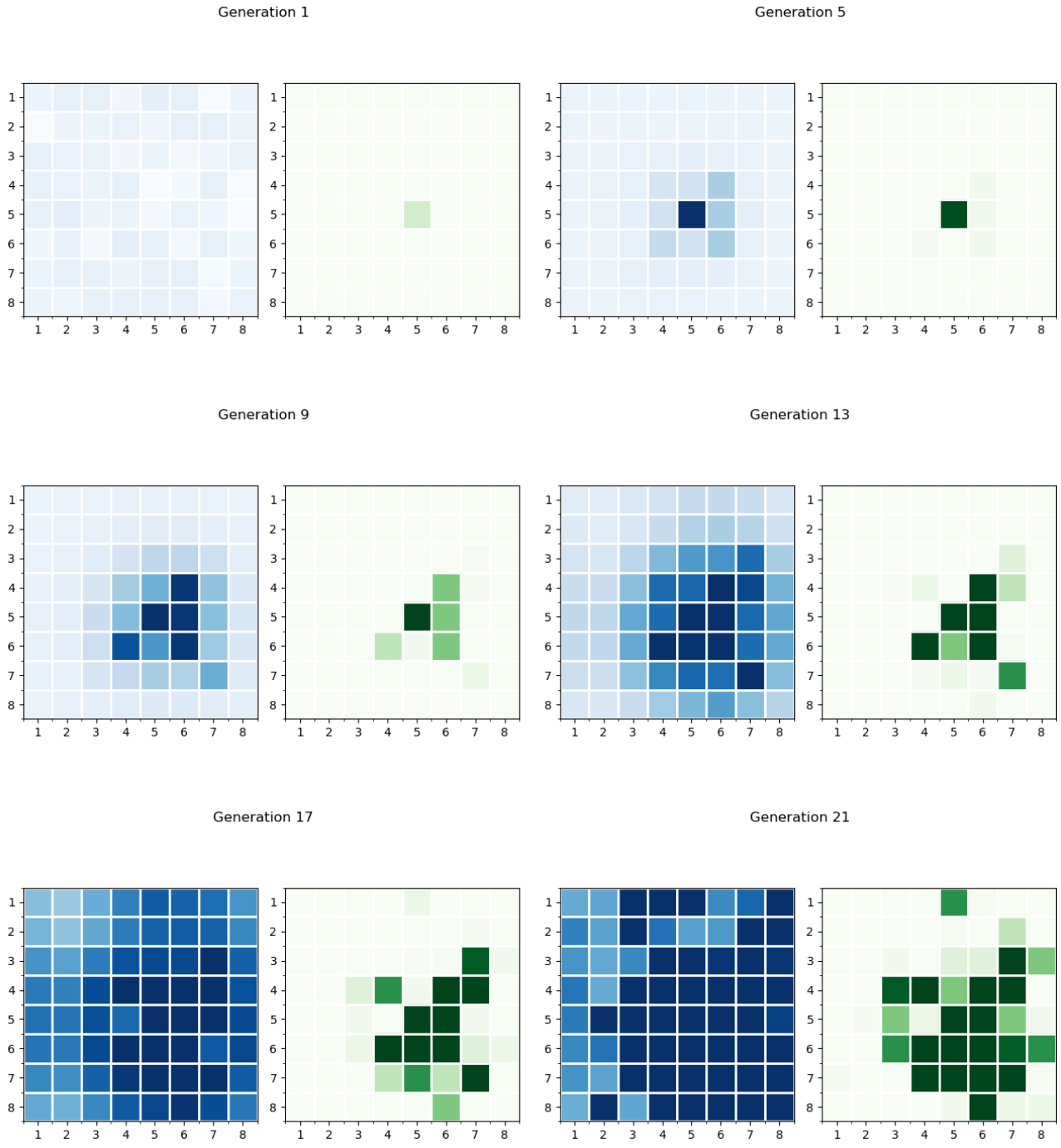

图 10: 生产者的增长和繁衍

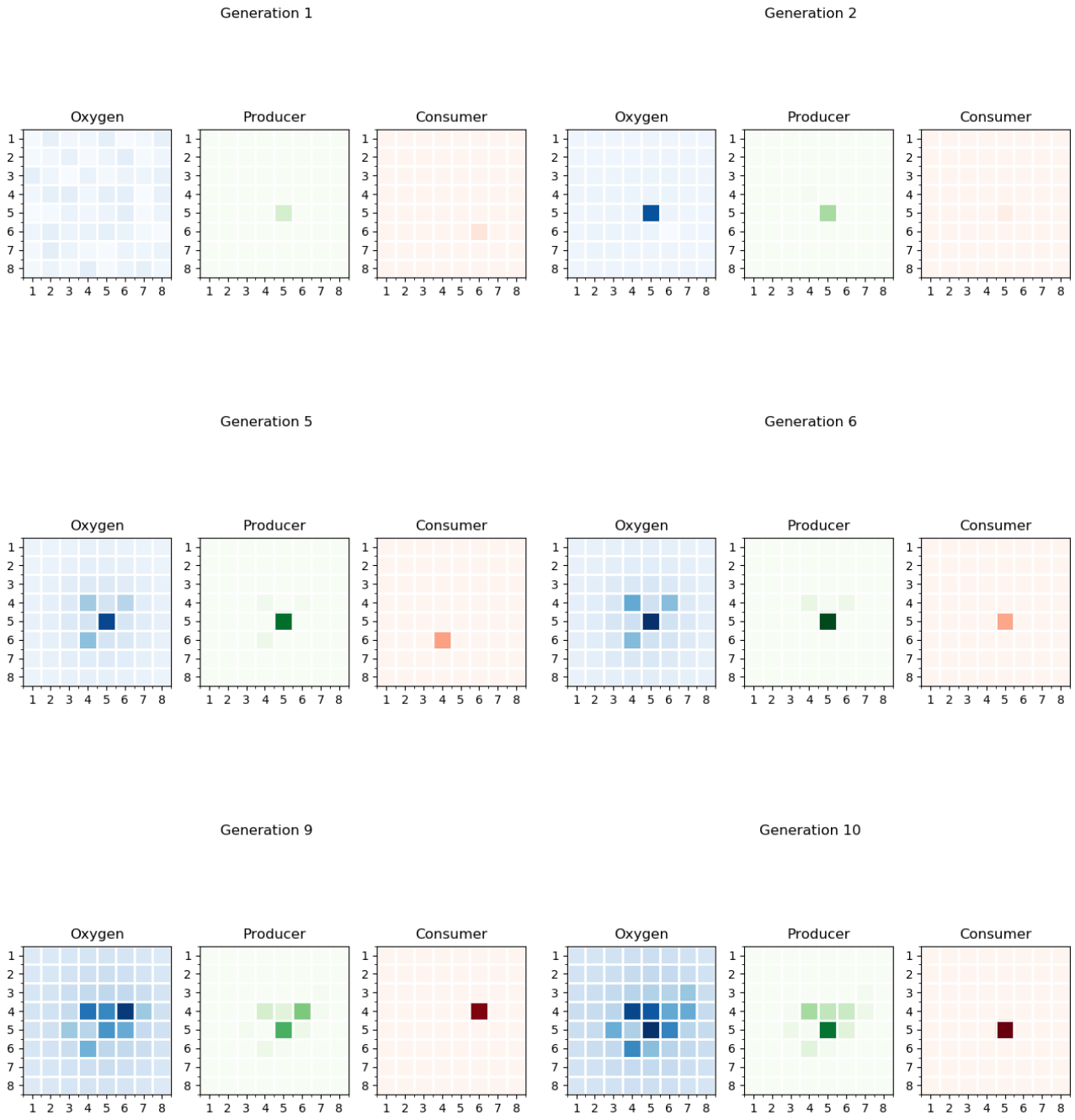

图 11: 消费者的呼吸、捕食和种群增长和移动

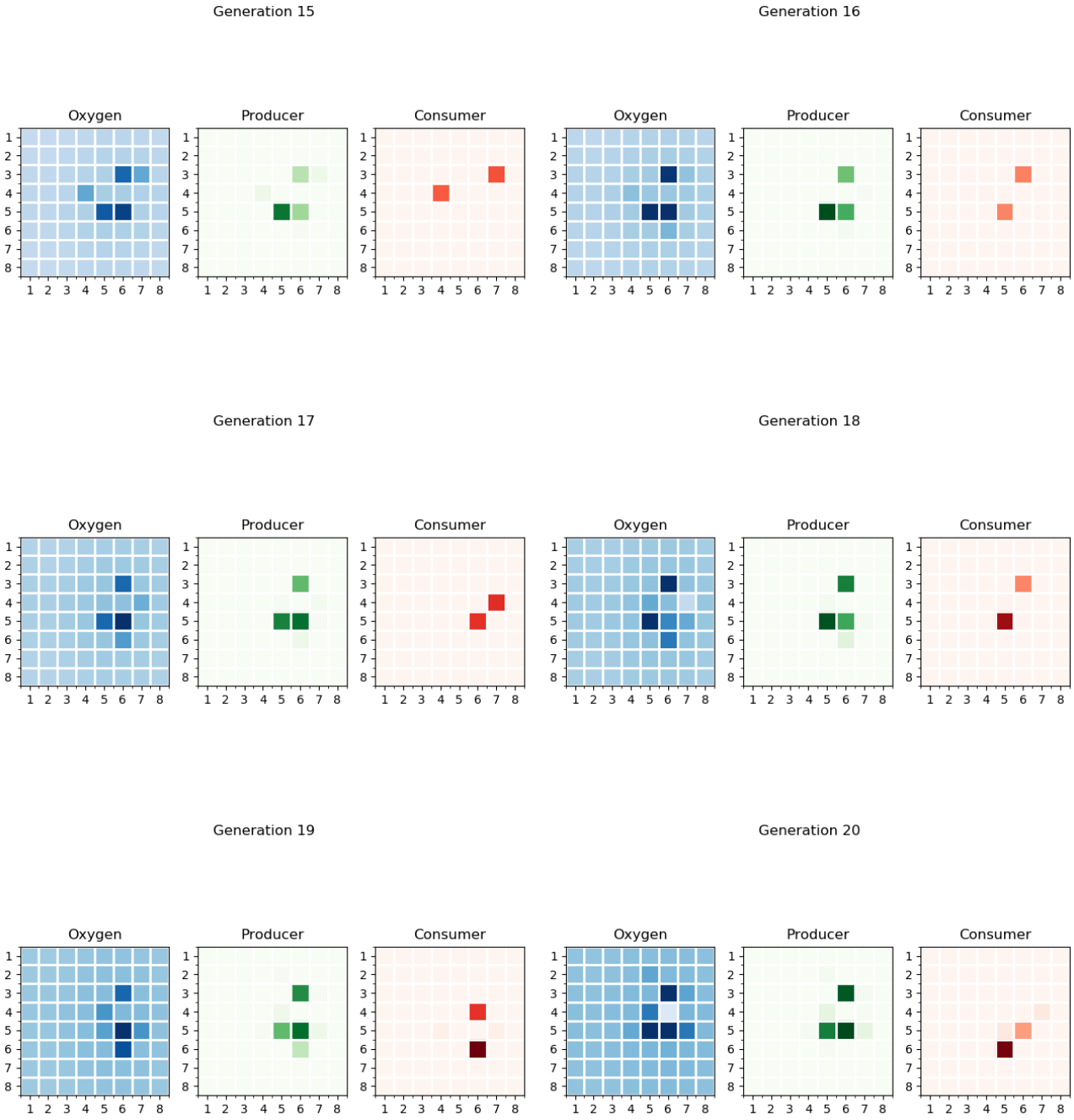

图 12: 具有完整行为的消费者

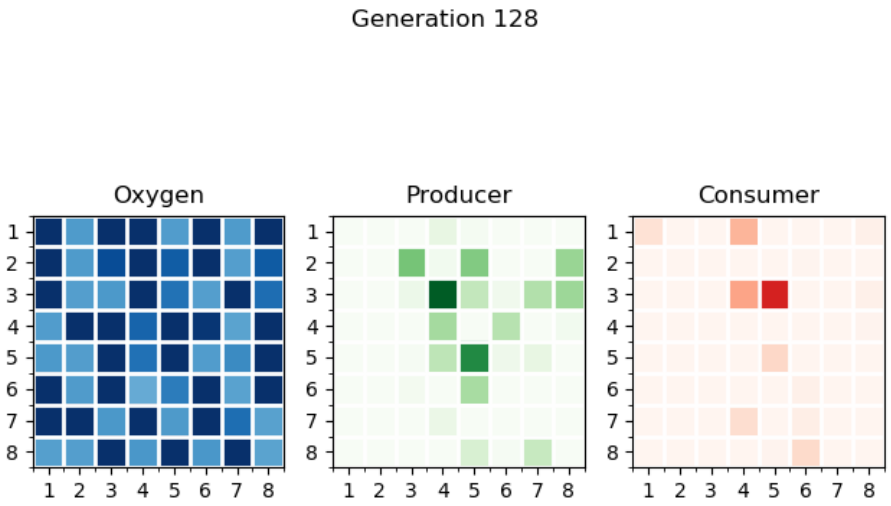

图 13: 弱光照条件（0.2）下迭代 128 次后的生态系统

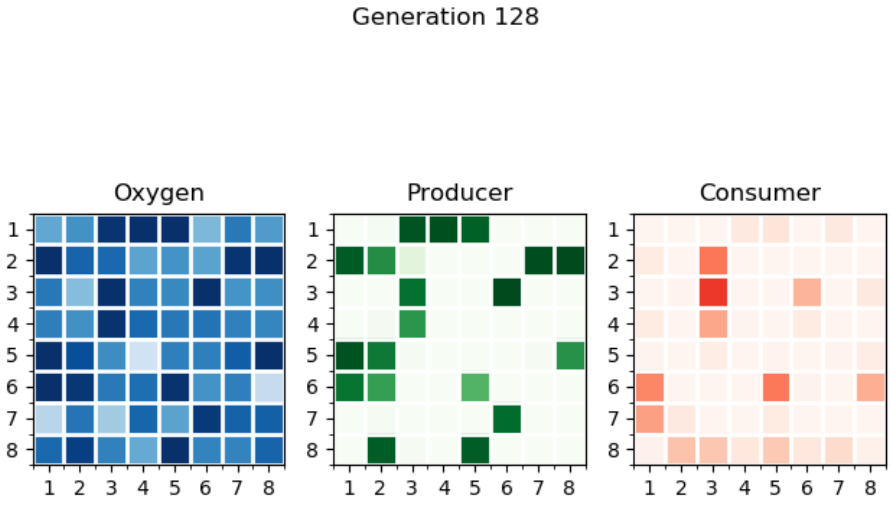

图 14: 中等光照条件（0.5）下迭代 128 次后的生态系统

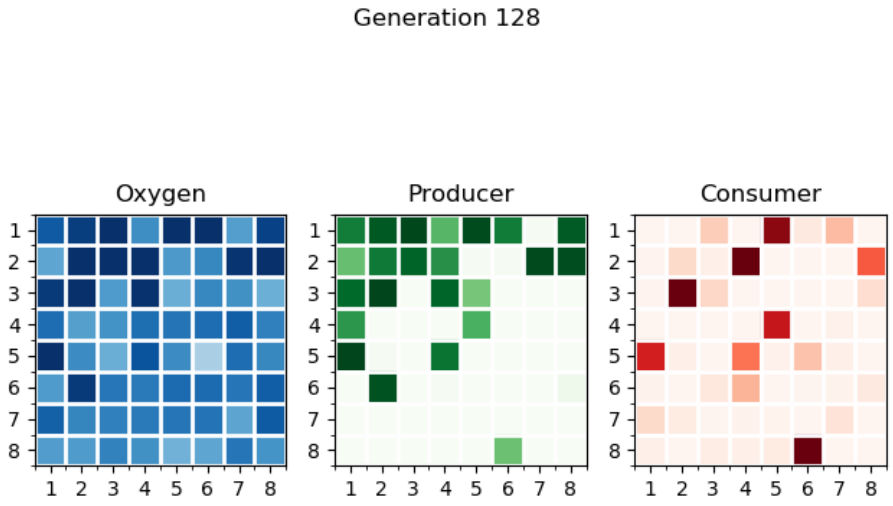

图 15: 强光照条件（0.8）下迭代 128 次后的生态系统

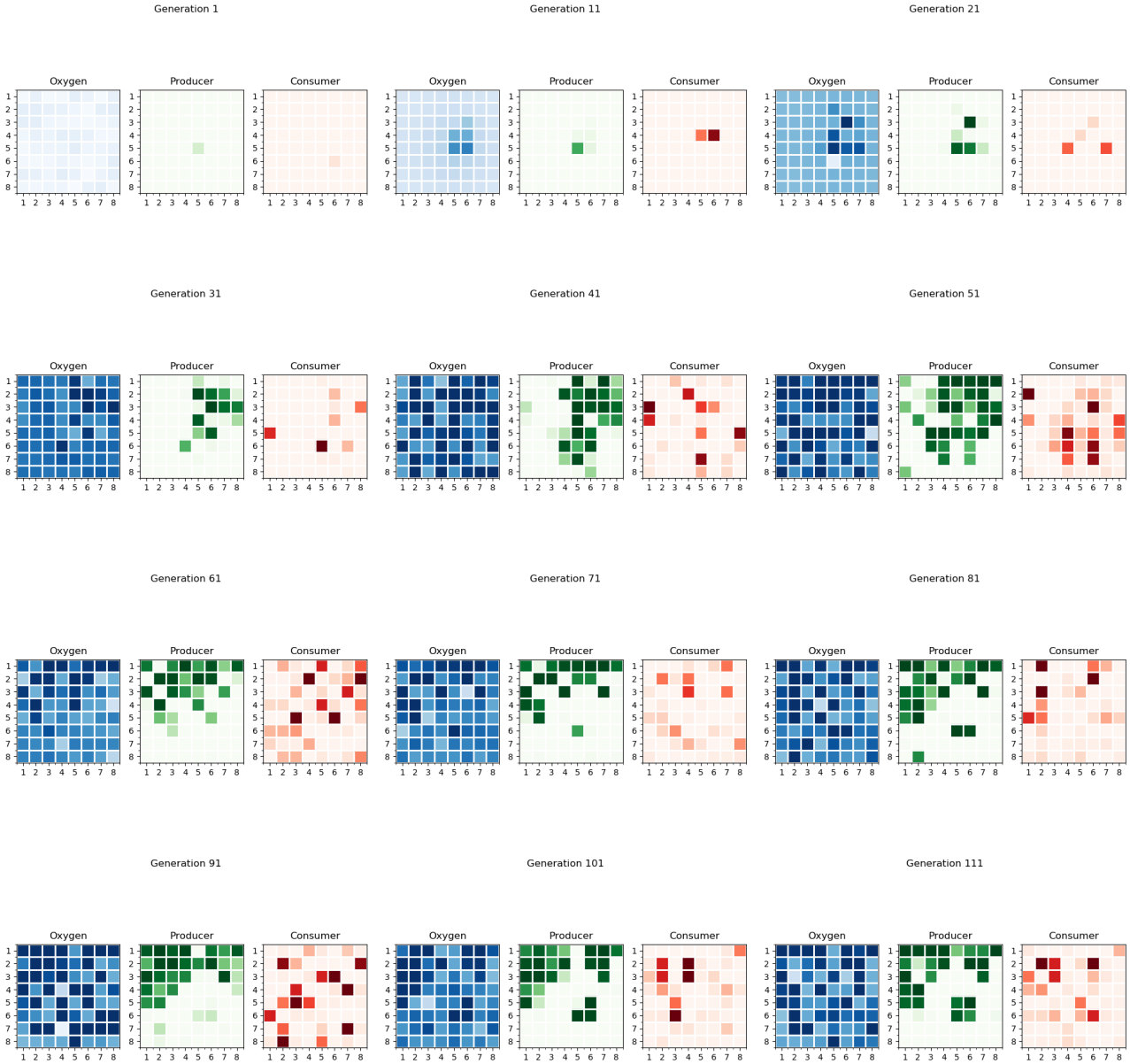

图 16: 一个生态系统的演化过程