Genetics of simulated bacteria colonies

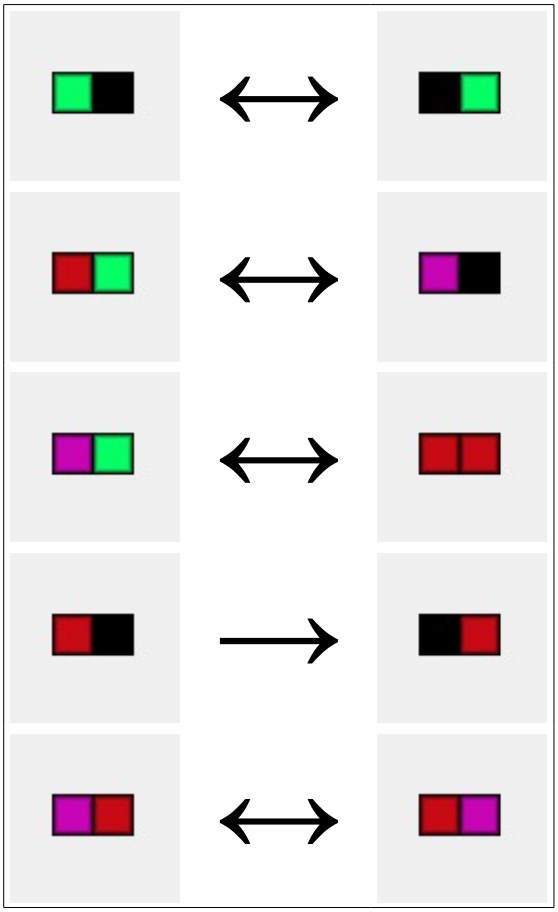
This is an explanation on how the “genetic bacteria” simulation works. First I will explain the mechanics of the board and the possible states of each cell, and then the functioning of several bacteria colonies in the simulation.

The simulation occurs in a square grid, just like in most cellular automatons. In this case it may be convenient to adjust the script so that the borders of the grid are connected in a toroidal map.

There are four possible states for each block in the grid: black (0), green (1), red (V) and pink (A). The first is just like empty space, the second plays the role of energy or a nutrient for the bacteria, the third is a living cell and the fourth is also alive, but it contains additional energy.

The blocks interact by pairs, so the first step for actualizing the position is to divide the grid into “dominoes”. This can be done in a great variety of ways, but, in order to produce a stochastic simulation and avoid too much complication, the way it is done in my programs is just choosing a predetermined number of random squares and one of its neighbors, and checking if we already put a domino in one of the squares, in which case we just don't pick that pair of squares.

When this process has concluded, we take a look at each domino. If the second square is alive (V,A) and the first one is not (0,1) we switch them. Then, for each combination of blocks there is (at most) one possible change. So all we have to do is decide whether or not to do it. The legal changes are:



The first of all occurs with probability ½, although this may be changed. The others depend on the ADN of the bacteria, which I will explain later.

Changes 1 and 4 can be interpreted as movement (of energy and bacteria), 2 as captation / liberation of energy, 3 from left to right as reproduction (mitosis) and 5 as transmission of energy among bacteria.

It is important to notice that this changes obey a nice conservative law: The quantity of “matter” in a given area remains constant under this changes.
¿How much matter is there in each block?
Well, if we take 0 as the matter in black void, and 1 as the matter in green energy, we can deduce that:

$$V+1=A(+0) \qquad (Change\,2)$$
$$A+1=V+V \qquad (Change\,3)$$
$$V+2=(V+1)+1=_2 A+1=_3 V+V$$
$$V=2; \quad A=3$$

Therefore, a bacteria colony can't grow indefinitely.

Now for the genome of the bacterias. In each case, the genetics should say whether or not a domino with a living cell (V, A) should undergo the (only) possible change. This decision may depend, of course, on the state of the living cell (V, A) and the state of the other cell in the domino (0, 1, V, A). Let's see some examples of bacteria in which the only parameters are this. The ADN looks like a bitstring of length 8, for example: `0110 1011` (the last bit doesn't matter, since the domino A-A has no change associated. This means that:

BIT	0	1	1	0	1	0	1	1
Cell 1	V	V	V	V	A	A	A	A
Cell 2	0	1	V	A	0	1	V	A
Change	4	2 (→)	3 (←)	5 (←)	2 (←)	3 (→)	5 (→)	(NaN)
Happens	Never	Always	Always	Never	Always	Never	Always	(NaN)

The most interesting parts of this kind of ADN are bits 1, 3, and 6.
Bit 1 regulates movement of 'V' cells. In this case, it is 0, so 'V' cells can't move in empty space.
Bit 6 and 3 act on cell “division” and the opposite change, let's call it “merging”. In this case, cells can merge, but not divide.

As these are the only changes that affect the number of live cells, we can conclude that a finite colony with this ADN will not grow at all, and in fact it will probably shrink with time.

Let's look at another ADN: `1001 1110`

It may seem that a colony with this ADN does tend to grow as time unravels: cells can move, divide and certainly not merge, so the number of live cells can't decrease. Again, that is wrong. Let's see which changes modify the number of pink (A) cells:

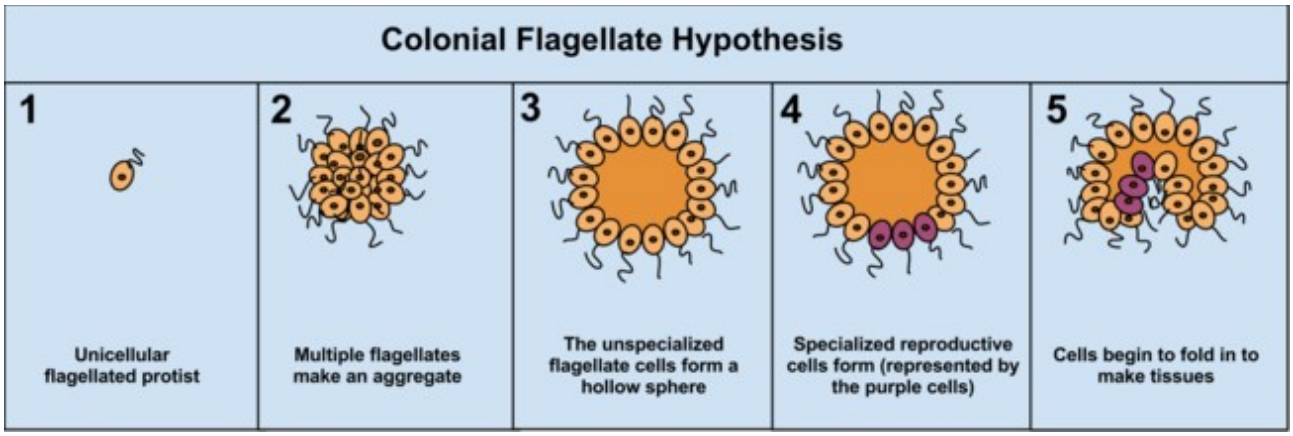
Change 2 (bits 2 and 5): Energy absorption / release. The cells can release energy, but not absorb it.
Change 3 (bits 3 and 6): Cell division / merging. The cells can divide but not merge.

In addition, there is not a single change that creates new A blocks. That means the number of them won't increase, and therefore only a finite number of cell divisions can be done. This fact, combined with the restriction on merging (bit 6) makes it a size-stable colony.

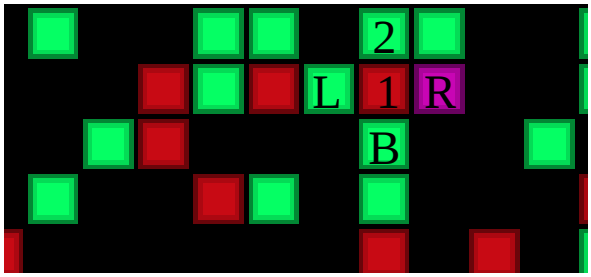
On the other hand, let's look at an ADN that *does* make the colony grow until the energy blocks are depleted. This is the bacteria `0101 0111`. As we can see, the cells can divide but not merge, and absorb energy but not release it (also change 5 allows energy to flow inside the colony). The colony does not just grow indefinitely when it has a source of energy; it is also unable to shrink. Besides, the bits 1 and 3 are 0, so every block must be adjacent to an already existing block. In we begin with a connected colony, the simulation will lead to another connected colony.

In general, one could classify every possible ADN as “dwindling”, “stable”, “growing” or “chaotic” (an example of the last is `1111 1111`). This is not important if it's the genetics what we are focusing on. Let's think about a more complicated problem: How can we can make these bacteria more intelligent? How could we make them act as a multicellular being? (without having more that one ADN).

Well, if there is an answer, it is not a simple one. According to Wikipedia, it took 35 million years for the first multicellular life forms to develop from unicellular beings. You may also know that in the human body, all cells share the same genome, and yet there are several kinds of cells and tissues. This can happen because cells have ways to “communicate” with each other using chemicals.



Our goal is to try to emulate that behavior expanding the DNA bitstring to take into account more parameters than just the cells inside the domino. For example, we could also take into account the three other cells that surround the first cell of the domino:

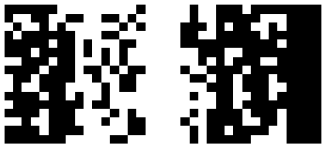


Now the DNA has a total of 5 dimensions: the first block of the domino (1) which can hold two values (V and A), the second one (2), L, B and R (the left, bottom and right blocks) which can hold four values each. Therefore, the ADN can be understood as a bitstring of length 512 (2·4·4·4·4) of a tensor of dimensions 2x4x4x4x4. Now, of course, the possibilities are many more.

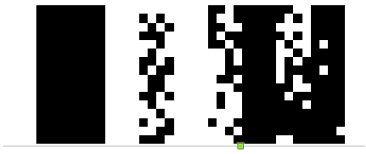
The first thing to take into account is that most of the ADN values are unused and unimportant. For example, the tensor element of indexes (1, 1, 1, 1, 1) makes reference to a pink A cell surrounded by green energy blocks. This is very unlikely, because green blocks become rare quickly and if a pink block has some green blocks nearby, it might divide before all of them are green.

On the other hand, some L-B-R combinations are more frequent, and this bits of the ADN are important. Among this are (0, 0, 0), (1, 0, 0), (0, 0, 1), (0, 0, V), etc... Now, how can we select the genomes that are more successful at colonizing the grid?

The best way seems to be just picking one of the copies of the most promising 8-bit ADNs, ignoring the values of L, B and R, and perhaps leaving random values for the least important bits. One way to visualize this is to split the 2x4x4x4x4 tensor into two 16x16 matrices, so that we concatenate the possible combinations of values of 2 and B in one dimension, and in the other, R and L. Let's see how a random ADN would look:



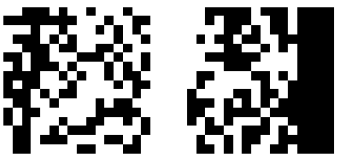
Now, for the manually optimized ADN that should invoke a quickly-growing colony:



Another possible approach for finding quickly-growing ADN is the one that nature uses: evolution. In order to implement what is called a genetic algorithm for colony size, we shall do the following:

1. Start with a set of 10 randomly-generated ADNs.
2. Generate 10 random boards with the same parameters, and put some A cells in the middle.
3. Let them run for a specified number of iterations one in each board.
4. After the simulation has ended, count the total number of A and V cells.
5. Repeat 2-4 two more times.
6. Select the 5 best ADNs (the ones that gave the most V & A), discard the rest and combine the 5 remaining ADNs by pairs to create 5 new “children” ADNs.
7. Go back to 2 and repeat the procces over and over.

This algorithm may seem slow and based on luck more than in genetics, but the results are surprising: after running it for 100 generations (100 repetitions of step 7) we got the ADN:



That beats our engineered ADN by a close margin!

Of course, the most interesting part is how we combine the ADNs of the “parents” to create the “child” ADN. What my algorithm did was basically go bit by bit and choose with equal probability between the corresponding bit of each of the “parents”. Additionally, there was a small chance that the bit was reversed in what could be understood as a “mutation”. This was done to avoid that all the ADNs ended being very similar and there was no chance of improving. I am glad to say that it worked very well.