Capturing hyper tumor growth behavior using genetic algorithms in a cellular automaton

Isabella Bicalho Frazeto

1.Context

This report is part of the final project for Programming and Algorithms. The python code for this at under the “final_project.py”. The supplementary files of interest are “compressed_archives.tar.gz”.

2.Introduction

A simple definition of cellular automaton: it is a group of cells in a grid that goes through the same changes using a discretized time to re-evaluate a grid base on the cell’s neighborhood. The only input the cell receives is its initial states. We can define the states in varying ways, but most of the time they are only two. In that case, zero represents a dead cell, and one a cell that is alive.

Since a genetic algorithm is a metaheuristic used to find a solution to optimization related problems, we used it to learn what set of rules we can implement in our automaton to generate the best solution. The basis of this algorithm is bioinspiration. It borrows evolution strategies to design algorithms to select the desired features. To summarize, this algorithm consists of a selection of DNA, typically referred to as chromosomes, based on a fitness function that gives a score of how this DNA performs in solving our problem. Using the score, we might select the best and mutate them or mate between themselves. Over a given number of generations, the selection system will optimize the solution best adapted to our problem.

I decided to put my own spin on this problem by trying to capture model growth behavior using cellular automata. First, this is not a new problem. There is extensive literature regarding this problem as reviewed by Nuttavut *et al.* Secondly, I will be not trying to capture all of its behaviors. Instead, I chose to focus on the probability of division using the Moore neighborhood, the cell death probability, and the probability of moving further into another state (more details about this in the following section).

My focus on this work is on Peto’s Paradox. In theory, large animals with a long life span, such as whales, should be more susceptible to cancer since they have more opportunities to initiate a tumor. Since they have a vast amount of cells, and their cells live for an extended period. However, this correlation does not exist (Caulien A. F & Maley C. C). At least, not in the wild.
Within a species, animal size is associated with increased cancer risk (Caulien A. F & Maley C. C).

Therefore, this suggests that large animals developed strategies overcame the greater risk of cancer. There are several hypotheses to describe how large animals and long-lived animals do it. The most relevant for our work is that they might have lower somatic mutation rates and the formation of hyperturmors.Hyperturmors are the idea that within a tumor, there are cells that are encouraged to cheat.

Hyperturmor cells take advantage of the structure build around tumors acting against the best interests of the tumor. In the end, it reduces its overall fitness, forcing it to reduce in size and even disappear (Caulien A. F & Maley C. C). In this case, a large long-lived organism would have more cancer, but tumors often evolve to hypertumors decreasing fitness and causing a negative correlation between case fatality rate and host body size (Crooper *et al.*).
Our goal in this work is to use a genetic algorithm to find the best parameters to describe hypertumor growth behavior using a cellular automaton.

3. Modelling

The following sections describe how I attempted to model hypertumor growth and create its genetic algorithm.

3.1 Hypertumor modeling

My initial idea was to capture this tumor behavior by modeling a “mutation threshold” to differentiate between normal cells, tumors, and hypertumors. However, this proves to be tricky to implement. Moreover, this causes the genetic algorithm to work with two distinct functions (the one that evaluates the system evolution and the other that causes the mutation threshold). So, I decided to model the problem as a system of three states where the third state evolution, hypertumor, is dependent on an additional parameter called mutation rate (in the code, this is the function update).

This change in modeling causes my algorithm to run faster. However, I cannot say precisely how much since I did not time it. Furthermore, it is the system now maps to three states but has infinitely many ones. The zero states are the normal cells, the ones are the tumor cells. The two or more are hyperturmors. For hypertumors, the state is not discrete, but instead, it is quantitative as a function of its mutation score.

3.2 Growth modeling

Since the evolution is hyper tumor growth-dependent, mutation rates must decide the overpopulation and underpopulation. In our model, overpopulation and underpopulation will lead to cell death for hypeturmors. It sounds counterintuitive at first, but it is not. Underpopulation means that the underlying tumor is no longer viable, so the hypertumor loses its support network and dies. Overpopulation means two things: competition is too high, or cells are unstable. In the first case, cells became not viable unless they are further mutated. In this sense, our cell cannot survive because it is not adapted to its environment. In the second case, it means that cells reach a threshold where it no longer lives. To simplify things, I used it as the mutation rate product with its respective parameters for tumor growth to find the parameters for the hypertumors.

However, this does not mean that hypertumors cannot divide since I kept open intervals, and I added one to the underpopulation obtained so we would have a range where cell division could occur. There is a catch here though if one cell that is a hypertumor multiplies, it is further mutated. I only used Moore Neighbourhood was to account for our problem.

3.3 DNA

I created a class called "Chromosome" to use as a unit. It is straightforward and contains four parameters: “overpop”, “underpop”, “reproduction”, “mutation_rate”. The first two are defined as the overpopulation threshold and underpopulation threshold, respectively. The third is the reproduction threshold for normal cells. In essence, this captures the rate at which normal cells turn to tumors. The last parameters were already discussed in previous sections.

I decided to create a class instead of using packages or a list because it would a more elegant and reproducible. I also think I chose it because I tend to make every subproblem into a function, a class, or an object. Probably, this is because when I started programming I used the Objected Oriented Programming Paradigm.

3.4 Crossing over and mutation

Both crossing over and mutation are functions in my code that take Chromosome class as a parameter. Crossing over takes two instances of the class and mates them. It just takes the first two values of the first instance with the last two of the second and vice versa. However, this function only returns one class instance. To add an element of stochasticity, only one of the crossings over returns.

Mutation function takes one Chromosome and draws one random value between zero and a given limit. I set the default value to be ten for no particular reason. I also did not change this value during my simulation.

3.5 Fitness

My function for fitness is rather a simple one. I decided that I wanted to have mostly normals cells and as few tumor cells as possible. Since our model is supposed to capture hypertumor cells as transient states, I also double-checked that it favored tumor cells instead of having hypertumors. The presence of hypertumor cells in the absence of tumor cells did not make sense under our framework, so that was penalized. Therefore, we can say the fitness comprises three terms: normal cells, tumor to hypertumors and tumors, and tumor cells to normal ones.

All those terms are pondered according to the weights (given by the user or the function default). This allows fitness to change easily by setting terms to zero if you do not want to depend on them. You could also use the sign of the number to change its meaning.

3.6 Selection and Population

Initially, I set the solution for only the best individuals. I will be referring to these best individuals as parents. However, this causes my system to quickly converged, which was nice, but I wanted to add a little bit of noise. I decided to create a genetic drift effect in the population by adding a random chromosome in the population pool. After the first generation, the population consists of the parents, offspring, mutated parents, and drift individuals.

3.7 Evolution

Early on, I decided I was not going to halt my simulation when I found the optimal solution because I also wanted to compare if my population would have some sort of “convergence” evolution. They were also trained in a new initial condition on the grid each time. My decision to do this was an attempt to reduce bias in my analysis.

4. Results

I ran a total of five trials that changed the parameters of my simulation. I will be discussing each one individually in the following sections.

4.1 First Trial

My first trial ran with a population of 10 individuals and generations. I selected the best four parents followed by two individuals coming from crossing over, two coming from mutations, and two from drift. For the fitness function, the weight of the normal as two, the hyper tumors weight as two, and the weight of the tumor weight as one. All other parameters ran as default.

In the end, the higher fitness was 3.959800020406081 My algorithm found the best parameters to be: 3 for “overpop”, 6 for “underpop”, 3 for “reproduction”, and “9” for mutation rate. The most adapted was the 8th individual for my population. Interestingly, this population shows cases of

convergence evolution. I did not keep track of all of the individuals, but I believe most of them would have similar fitness values judging by the similarities of the results. (Figure 1).

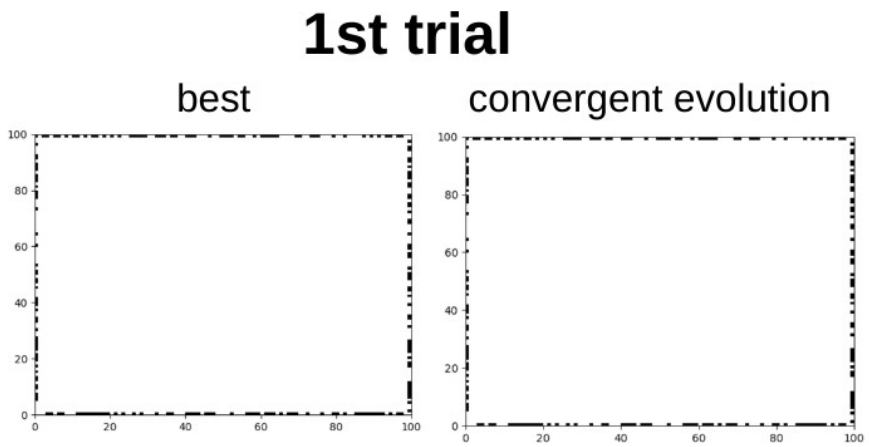

Figure 1. Results for the first trial. Both the best and the other evolution had similar results.

4.2. Second Trial

My second trial ran with a population of 50 generations and 20 individuals. I selected the best five parents with five individuals coming from crossing over, five coming from mutations, and five coming from drift. For the weight of the fitness function the weight of the normal as two, the weight of the hyper tumors as two, and the weight of the tumor as one. All other parameters ran as default. Interestingly, the second trial had the exact same fitness value (3.959800020406081) as the first one. The best was the 14th individual of the population. The values for “overpop”, “underpop”, “reproduction”, and “mutation_rate” were 8, 7, 7, and 0, respectively. I also found a similar solution to this one, but with smaller fitness values (Figure 2).

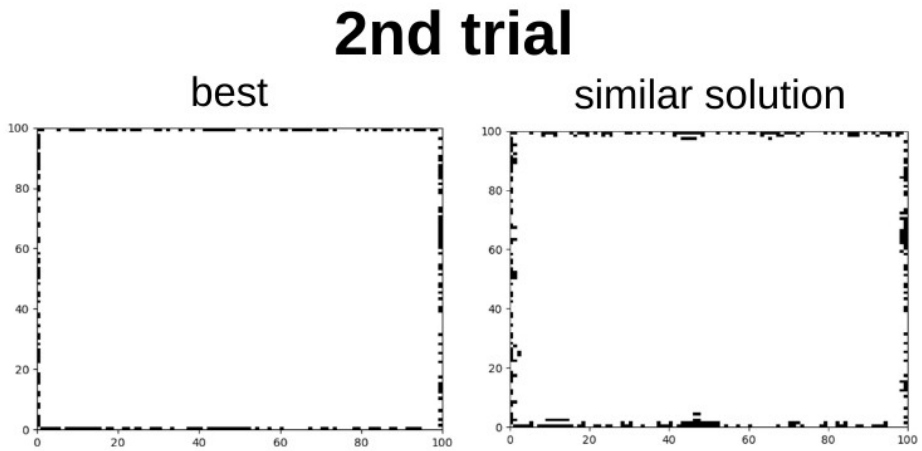

Figure 2. Results for the second trial. The best result is simular to other similar solution.

4.3 Third Trial

My third trial ran with a population of 50 generations and 20 individuals. I selected the best five parents with five individuals coming from crossing over, five coming from mutations, and five coming from drift. For the weight of the fitness function the weight of the normal as two, the weight

of the hyper tumors as one, and the weight of the weight tumor as one. All other parameters ran as default.

For the third trial, we have, yet again, the same fitness value as the first and second ones. The values for “overpop”, “underpop”, “reproduction”, and “mutation_rate” were 2, 4, 5, and 4, respectively. The best individual was the 18th individual. Even though I was not keeping track of how fast the individuals arrived at the final solutions, there seems to be a correlation since some other individuals arrived at the same solution slower but did not have the best fitness as shown in figure two.

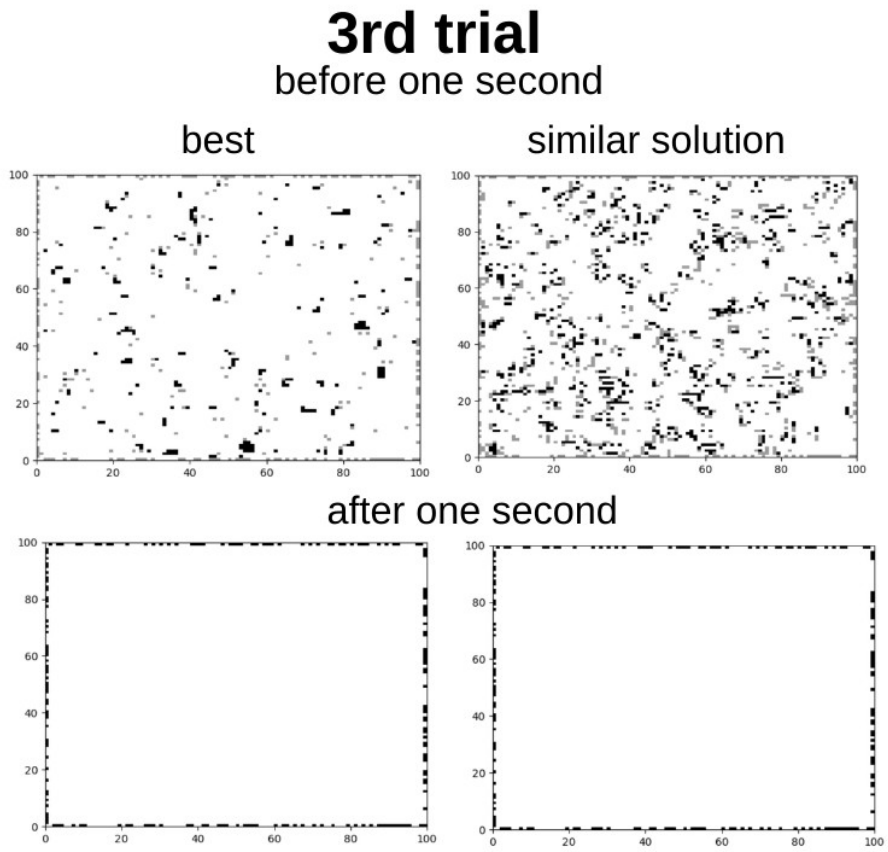

Figure 3. The best solution arrived seems to arrive faster to the solution even though the end result is the same.

4.4. Fourth Trial

For the fourth trial, I run with the same parameters and the third trial, but I changed the fitness function. Since my algorithms do not update the borders I was not previously counting cells in it. I decided to count it for this trial and the next. This, of course, causes the fitness function to fluctuate more. The values for “overpop”, “underpop”, “reproduction”, and “mutation_rate” were 4, 4, 5, and 7, respectively. The best individual was the 16th individual. The absolute fitness value was 212.0. During these trials, it is possible to see that the hypertumors die out in the tumor cells' absence (Figure 4). A lot of individuals had similar convergent evolutions.

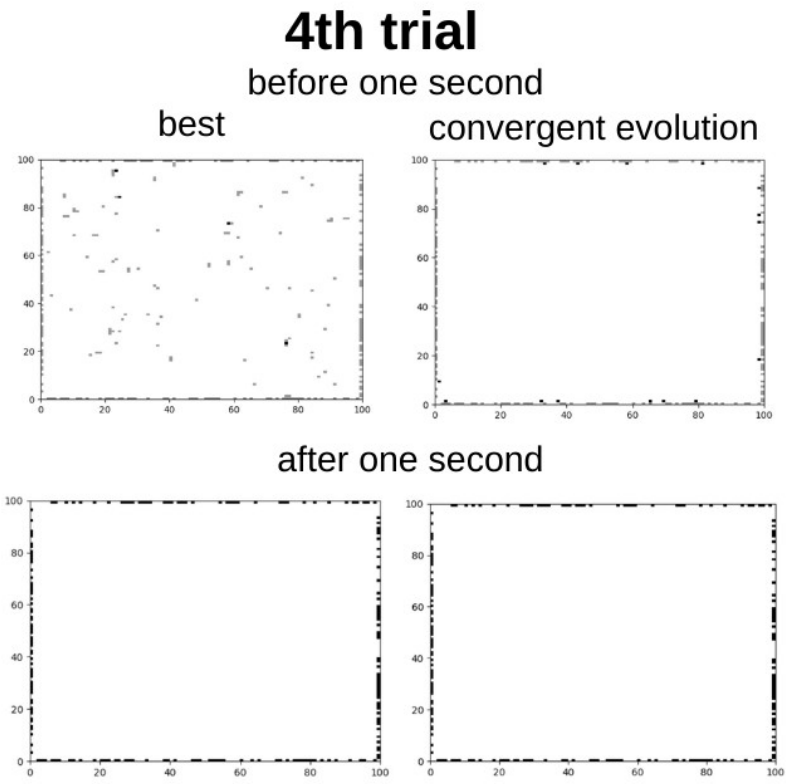

Figure 4. Similarly and the third trial, the fourth trial best solution seems to be faster than other solution. However most of the simulations arrived at the same solution.

4.5 Fifth Trial

My fifth trial ran with a population of 50 generations and 20 individuals. Again, I selected the best five parents with five individuals coming from crossing over, five coming from mutations, and five coming from drift. For the fitness function, the weight of the normal as two, the weight of the hyper tumors as one, and the weight of the weight tumor as two. Similar to the fourth trial, I counted the cells at the borders. All other parameters ran as default.

The best absolute fitness value was 216.01122334455667 for the 19th individual. The values for “overpop” was 7, for “underpop” was 9, for “reproduction” it was 2. Finally, the “mutation_rate” was 2. Similar to the third trial, they were a lot of convergent evolution. There is one individual who I thought the behavior was very intriguing, individual 4th. The comparison between the best individual and the individual 4th is in Figure 5.

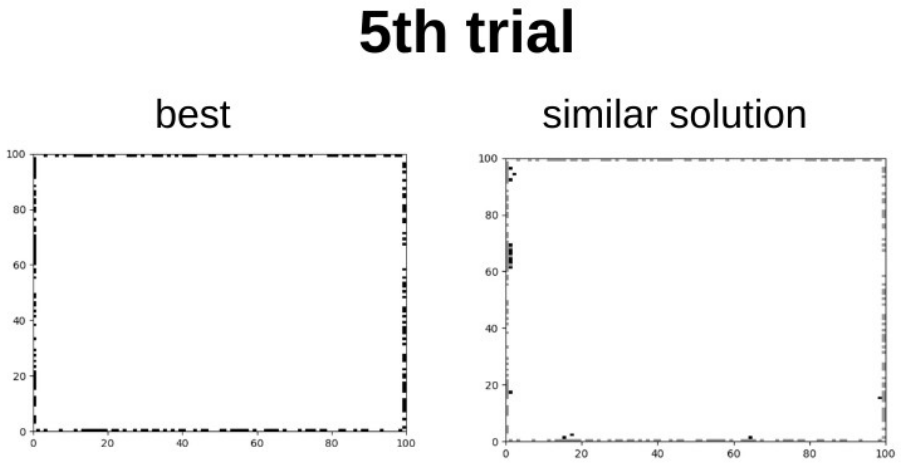

Figure 5. The best result compared to a simular solution that I found interesting.

5. Discussion

Overall, all trials came up with different solutions, but similar results. From the biological point of view, this is fascinating. I think it would be fruitful to study how this evolution behaves before the final solution and study the relationship between model parameters. The second trial had high values for overpopulation and underpopulation but zero as mutation rate. Similarly to the second simulation was also number five. Others, such as number one and four, had very high mutation rate values but low underpopulation and overpopulation values. I wondered if uncovering this, could provide further insights into how large long-lived animals manage to have fewer tumors than expected.

From the programming point of view, I believe I created a very naïve model. There is nothing exceptionally difficult in my code or elaborate in my code. To put it plainly, it works. However, I think it can improve. For instance, I should treat the borders to be able to change in the future. I also wanted to add checks for objects in the class chromosome, as it is it does not check the type of initial values.

I also put constraints around the initial parameters (zero to ten) because my computer would not run such long simulations without it. It would be insightful to explore besides my hardware limitations.

Besides that, I would re-run simulations number four and five. I should have changed the tumor weight value for a negative number so the fitness function would be positive. Hence me using the absolute value to describe them. It did not have a huge impact in the results or the selection (Figure 4 and 5), but it is just wrong. However, this did capture the idea that absolute value of fitness went down as the simulation progressed. Another possible change in the fitness function would be to change the numor of tumor cells to a relative number instead of a absolute one.

I also believe this is a relatively easy problem to solve despite the somewhat complicated biology behind it. During my test runs, I could see that the solution would emerge at the first draw of the population, and it would stay until the end of the simulation. In conclusion, it seems that this problem has multiple optimal solutions.

6. References

J. Poleszczuk and H. Enderling, "A High-Performance Cellular Automaton Model of Tumor Growth with Dynamically Growing Domains,"Applied Mathematics, Vol. 5 No. 1, 2014, pp. 144-152. doi:10.4236/am.2014.51017.
Boondirek, Ankana & Triampo, Wannapong & Nuttavut, Narin. A Review of Cellular Automata Models of Tumor Growth. 2010. International Mathematical Forum. 5. 3023-3029.
Aleah F. Caulin and Carlo C. Maley. Peto's Paradox: evolution's prescription for cancer prevention. Vol 26, Issue 4, 2011.
Nagy JD, Victor EM, Cropper JH. Why don't all whales have cancer? A novel hypothesis resolving Peto's paradox. Integr Comp Biol. 2007 Aug;47(2):317-28. doi: 10.1093/icb/icm062. Epub 2007 Jun 28. PMID: 21672841.