A simplified model of bacteria antibiotic resistance focusing on membrane permeability

Ana Sofia Almeida[1,2], Carolina Vidal[1,3]

[1] *BBC Masters, Department of Informatics, Faculty of Science, University of Lisbon*
[2] *asofia.almeida1@gmail.com*
[3] *carolina.m.vidal@hotmail.com*

KEYWORDS

Bacteria Resistance
Membrane Permeability
Genetic Algorithm
Simulation

ABSTRACT

In this project a Genetic Algorithm (GA) was designed and implemented in order to determine the bacterial growth in response to antibiotic resistance, based on the permeability of the bacteria membrane. The GA was designed using Python Programming Language. Bacterial growth was conducted by cloning bacteria in an environment that simulates the presence of an antibiotic. The simulation results show that the biological principles of bacteria and their antibiotic resistance mechanism were correctly designed and implemented.

1 Introduction

Antimicrobial Resistance (AMR), refers to the ability of a microbe to resist the effects of drugs that they have previously been exposed to. These infections break out regardless of a nation's geographical conditions or socioeconomic status, being one of the world's most persistent public health concerns.[1] As a result of drug resistance, antibiotics and other antimicrobial medicines become ineffective and infections become increasingly difficult or impossible to treat, increasing the risk of disease. AMR occurs naturally over time, usually through genetic changes however, one of the main drivers of antimicrobial resistance include the misuse and overuse of antimicrobials.[1]

Gram-negative bacteria, in particular, have significant intrinsic resistance to many classes of antibiotics owing to their outer membrane (OM) which imposes a physical barrier that limits the range of molecules that are taken up into the bacteria.[2,3,4] This has been recognized as a contributing factor to the difficulty in finding potent new antibacterial agents acting against these organisms.[5,6] This barrier comprises a lipid bilayer that is impermeable to large, charged molecules. There are two different aspects to transport systems across the bacterial membrane — influx and efflux. Influx is largely controlled by porins, which are water-filled open channels that span the outer membrane and allow the passive penetration of hydrophilic molecules.[2,7,8] Many antibiotics acting against Gram-negative bacteria are polar molecules and most exploit the water-filled channels to enter the cell.[3,9] The interactions between porins and antibiotics have been studied using electrophysiology and all-atom molecular dynamics simulations to provide a molecular description of the translocation events.[10,11] Previous efforts have been made to construct a quantitative model of drug accumulation into E. coli cells considering both influx and efflux.[12,13] This pictures porins' function as non-specific channels for the passive diffusion of drugs across the OM and steady-state intracellular drug concentrations are balanced by the specificity and efficiency of transmembrane efflux pumps.[12,14]

The control of bacterial membrane permeability is a complex process that is tightly regulated by a complex network of systems that sense and respond to osmotic shock, pH, temperature, antibiotics and chemical stress.[15,16] Documenting the emergence or acquisition of antibiotic resistance mechanisms during antibiotherapy highlights the efficiency of the bacterial adaptive response. In this respect, decreased membrane permeability reduces the internal accumulation of antibiotics, therefore allowing time for the development of further resistance mechanisms, such as target modification or drug inactivation. This information is of importance for understanding the bacterial resistance that is triggered by the modification of membrane permeability and for the development of new antibiotherapy strategies. Deciphering antibiotic translocation through porins at the molecular level is crucial for understanding the correlation between influx and antibiotic activities in bacteria.

Bioinformatic or computational biology approaches to bacteria and antibiotic resistance will provide more efficient methods for conducting antibiotic resistance research. There are studies for predicting and estimating the results of antibiotic resistance using simulation methods. These include modeling

antibiotic resistance or modeling the biological systems of bacteria and their interactions with antibiotics.

2 Approach

The goal of this project was to develop a simulation, using a Genetic Algorithm in order to observe, from a biological perspective, the growth of a population of bacteria in an environment containing an antibiotic. The idea was based on a simple criterion, the bacteria membrane permeability. Each bacteria (individual) is represented by a binary string of length 20, where each one represents the presence of pore and each zero represents the lack of a pore. Consequently, the best individual would be represented by a string containing only zeros, which is the equivalent of saying that the bacteria does not possess any pores in its membrane. Since a reduce number of pores lead to a decrease of the membrane permeability, we expect that the antibiotic present in the environment will have more difficulty to diffuse across bacteria membrane, allowing the bacteria to gain resistance against it. Once this happens, bacteria can now reproduce itself more, because the selection pressure caused by the antibiotic has a minimal effect and thus, we expect that the bacteria fitness will increase over time and that this bacteria will reproduce itself more, increasing its number on the population. It is important mention that bacteria do not reproduce sexually but asexually, meaning that each bacteria originates a copy of itself (clone), generating only one offspring each cycle. Therefore, the crossover step was not taken into account.

In order to achieve this, the **initial population** was set to hundred individuals so it wouldn't be too big and eventually slow down our program, nor too small and in this case lack diversity. Each individual is created by generating a list of a fixed length (20), with a random number of zeros and ones and then randomly shuffling it. All these lists are then joined into a single one, that represents the initial population itself.

To measure **fitness**, the best possible individual, created as our reference, was developed by generating a list with only zeros (no pores) which embodies an individual with the highest antibiotic resistance, accordingly to our criterion. Each element of an individual is then compared with each element of the best possible individual (reference) and if they are equal, that is, have the same number in the same position, then the temporary fitness score is augmented by one. After this process, the final fitness score is calculated according to the following equation: $fitness\ score = \frac{temporary\ fitness\ score}{maximum\ number\ of\ pores} * 100$

Selection is deterministic and based on the fitness measure, selecting the individual whose fitness score meets the following requirement: $fitness\ score > 90\%$. Therefore the selection is elitist, selecting only the best individuals from the joint set of parents and offspring.

In the **reproduction** step, all the selected individuals are cloned. The fitness score of each individual in the population is calculated, and if it matches the criteria is selected for cloning. The select individuals are then added to the population list.

The **mutation** process is probabilistic, with a rate of 0.002, and independent of the individual's fitness. To apply the mutation randomly, it is first created an array, with a random probability of each individual suffering a mutation. Then a Boolean array is created by verifying if this probability is below the mutation rate (0.002); afterwards if the Boolean value reveals to be true, the individual is mutated.

The process involving all of these steps, is repeated for a determined number of generations.

To analyze our data, we keep track of the mean fitness score in each generation, that is done by saving each individual's score into a temporary list that is cleaned in each generation. In each generation, we calculate the mean value of this list and save it. Later, the data is plotted to better visualize how the mean fitness of the population progresses through the generations.

To compare how much each individual clones itself, as it depends on each individual fitness, all the unique genotypes are found and the number of times each one repeated itself on the population is counted. As there are many genotypes, the mean number of clones for each genotype was calculated and all of those who were below average were excluded, so our plot could be more understandable. With the resulting group, a bar plot was created that showcases the number of clones per genotype.

3 Implementation

The project was implemented with python based on a simple GA already constructed[17] and can be access through: https://github.com/cmv-project/va_project/. It is also in the same file as is this report.

When running our code, it is possible to specify through input the maximum number of pores, the population size, the number of generations and the mutation rate. As we used a lot of random functions, even if the parameters are the same it most likely will give different results each run. So, we also provided some csv files with the genotype of the initial and the final population, as well as individuals per generation and the mean score, generated in one of the runs of the GA with the following parameters: maximum generations=10, number of individuals=100, mutation rate = 0.002 and which we used as our reference latter in the results. In the program commentary, we provide code lines to use our example data files.

The program is composed of only one file (va_project.py) with multiple functions, being the most important ones already explained, above, in the approach.

Running the algorithm yields three plots, one with the mean fitness per generation, one with the highest fitness at each generation and other with the number of clone copies.

4 Results

Initially, the program was run with the following parameters: maximum generations=10, number of individuals=100, mutation rate = 0.002. The result we obtained in one of the runs is illustrated in the Figure 1, where the best individual is represented by index 9 and it was cloned approximately 3000 times. This individual corresponded to a string of zeros only. We can also see that this individual was already present in the initial population (Fig.1.1). The mean fitness of the population increases through the generations reaching the 90% in the last one, which indicates that the GA is working in order to achieve the best solution possible.

We decided to assess the effects of different parameters by changing each one of the three mentioned above, one at a time. When the mutation rate is altered to 0.1, the mean fitness of a population decreases at a certain point. However, in this case the GA managed to recover and by the 3rd generation the fitness starts to increase again but by the end of the 10th generation the medium fitness score only reaches approximately 50%. This occurs because the mutation rate is too high and becomes destructive leading to the loss of the best individuals, even if they exist in a high number and consequently converging to a local optimal solution.

Changing the initial population size to 50 and 200 results in similar plots of the mean fitness score, which increases with the generations, reaching 90%. (Fig.3). However, the number of genotypes generating clones were way higher when the initial population was set to 200 individuals in comparison with the run with 50 initial individuals and consequently it is notable the lack of diversity in the genotypes in the population composed of initially 50 individuals (Fig.3.1).

Lastly, when the number of generations was altered to 20 generations, we noticed that the fitness score seems to stabilize in the 15th generation. Showing once again that the GA is effective.

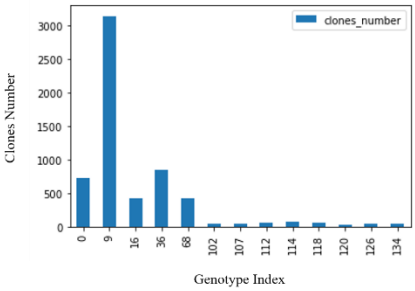

Figure 1 - Number of clones per genotype excluding those below clone number average. *Maximum generations=10, number of individuals=100, mutation rate = 0.002*

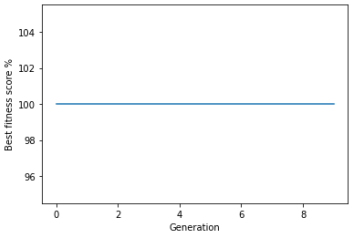

Figure 2.1 – Best fitness score in each generation

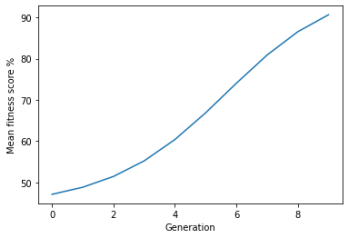

Figure 3.2 – Mean fitness score in each generation

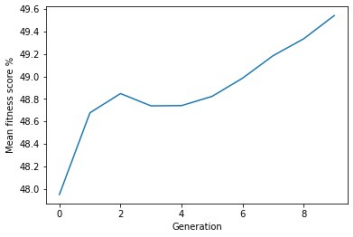

Figure 2 – Mean fitness score in each generation. *Maximum generations=10, number of individuals=100, mutation rate = 0.1*

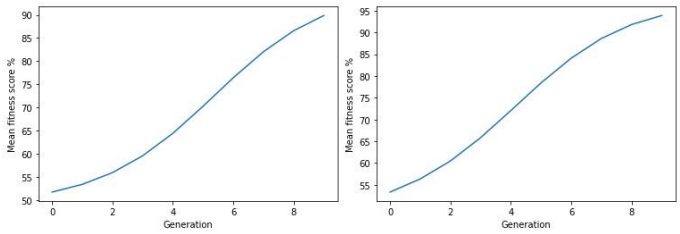

Figure 3– Mean fitness score in each generation.
Left: *Maximum generations=10, number of individuals=50, mutation rate = 0.002*
Right: Maximum generations=10, number of individuals=200, mutation rate = 0.002

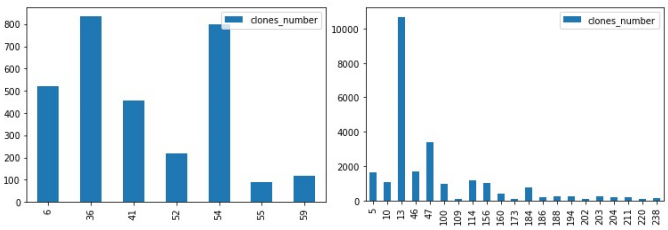

Figure 3.1 – Mean fitness score in each generation.
Left: *Maximum generations=10, number of individuals=50, mutation rate = 0.002*
Right: Maximum generations=10, number of individuals=200, mutation rate = 0.002

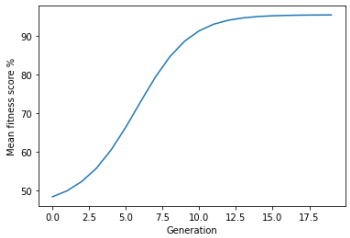

Figure 4 – Mean fitness score in each generation. *Maximum generations=20, number of individuals=100, mutation rate = 0.002*

5 Comments

It is important to highlight that in this project we designed a very simple model of a bacteria and its environment. We believe that the biggest problem facing this model is that fact that we are only considering that the decrease in the number of pores will increase the bacteria fitness by diminishing the membrane permeability. However, the control of bacterial membrane permeability is regulated by a complex system that sense and respond to different factors making pores an essential component of the outer membrane of bacteria, responsible for maintaining its homeostasis.

To achieve a more complex model several other aspects could be taken into account, such as defining an initial environment with a set of different parameters, such as food and different antibiotic concentration. It would also be interesting to test other aspects that leads to an increased resistance to antibiotics, such as, the structure of a key molecule that the antibiotic targets, rendering it ineffective. Or the production of a chemical that destroys the antibacterial properties of the drug. Once it emerges, antibiotic resistance can jump from one species of bacteria to another which makes it especially dangerous. Microorganisms naturally exchange genetic material in a process called horizontal gene transfer – either by close contact or by forming a sort of bridge between each other. Therefore, it would also be interesting to simulate the interactions between different bacteria.

Notwithstanding, our simulation results showed that the biological structure of bacteria, number of pores, and their antibiotic resistance mechanisms, in this case the membrane permeability, were correctly designed and implemented, being in accordance with the expected results. The presence of the antibiotic in the environment starts to favor the selection of wild-type strains that bear mutations conferring resistance to the antibiotic, and in time, antibiotic-resistant strains evolve from the wild-type population. In a natural competitive race for survival, the wild-type strain cannot survive in the presence of the antibiotic, however the resistant strain is perfectly capable of surviving the antibiotic and continue to reproduce itself.

6 References

1. https://www.who.int/news-room/fact-sheets/detail/antibiotic-resistance (last assessed in 12/01/2021)
2. Nikaido, H. (2003). Molecular basis of bacterial outer membrane permeability revisited. *Microbiology and molecular biology reviews*, *67*(4), 593-656.
3. Pagès, J. M., James, C. E., & Winterhalter, M. (2008). The porin and the permeating antibiotic: a selective diffusion barrier in Gram-negative bacteria. *Nature Reviews Microbiology*, *6*(12), 893-903.
4. Masi, M., Réfregiers, M., Pos, K. M., & Pagès, J. M. (2017). Mechanisms of envelope permeability and antibiotic influx and efflux in Gram-negative bacteria. *Nature microbiology*, *2*(3), 1-7.
5. Tommasi, R., Brown, D. G., Walkup, G. K., Manchester, J. I., & Miller, A. A. (2015). ESKAPEing the labyrinth of antibacterial discovery. *Nature reviews Drug discovery*, *14*(8), 529-542.
6. Brown, E. D., & Wright, G. D. (2016). Antibacterial drug discovery in the resistance era. *Nature*, *529*(7586), 336-343.
7. Delcour, A. H. (2003). Solute uptake through general porins. *Front. Biosci*, *8*, 1055-1071.
8. Schulz, G. E. (2002). The structure of bacterial outer membrane proteins. *Biochimica et Biophysica Acta (BBA)-Biomembranes*, *1565*(2), 308-317.
9. Masi, M., Réfregiers, M., Pos, K. M., & Pagès, J. M. (2017). Mechanisms of envelope permeability and antibiotic influx and efflux in Gram-negative bacteria. *Nature microbiology*, *2*(3), 1-7.
10. Winterhalter, M., & Ceccarelli, M. (2015). Physical methods to quantify small antibiotic molecules uptake into Gram-negative bacteria. *European Journal of Pharmaceutics and Biopharmaceutics*, *95*, 63-67.
11. Bajaj, H., Scorciapino, M. A., Moynié, L., Page, M. G., Naismith, J. H., Ceccarelli, M., & Winterhalter, M. (2016). Molecular basis of filtering carbapenems by porins from β-lactam-resistant clinical strains of Escherichia coli. *Journal of Biological Chemistry*, *291*(6), 2837-2847.
12. Westfall, D. A., Krishnamoorthy, G., Wolloscheck, D., Sarkar, R., Zgurskaya, H. I., & Rybenkov, V. V. (2017). Bifurcation kinetics of drug uptake by Gram-negative bacteria. *PLoS One*, *12*(9), e0184671.
13. Nichols, W. W. (2017). Modeling the kinetics of the permeation of antibacterial agents into growing bacteria and its interplay with efflux. *Antimicrobial agents and chemotherapy*, *61*(10).
14. Lim, S. P., & Nikaido, H. (2010). Kinetic parameters of efflux of penicillins by the multidrug efflux transporter AcrAB-TolC of Escherichia coli. *Antimicrobial agents and chemotherapy*, *54*(5), 1800-1806.
15. Valentin-Hansen, P., Johansen, J., & Rasmussen, A. A. (2007). Small RNAs controlling outer membrane porins. *Current opinion in microbiology*, *10*(2), 152-155.
16. Viveiros, M., Dupont, M., Rodrigues, L., Couto, I., Davin-Regli, A., Martins, M., ... & Amaral, L. (2007). Antibiotic stress, genetic response and altered permeability of E. coli. *PloS one*, *2*(4), e365.
17. https://pythonhealthcare.org/2018/10/01/94-genetic-algorithms-a-simple-genetic-algorithm/ (last assessed in 12/01/2021)