POLITECNICO DI TORINO

Master's Degree in Computer Engineering

Master's Degree Thesis

Evolutionary Computation techniques for the analysis of Antibiotic Resistance in bacterial infectants

Supervisors

Giovanni SQUILLERO

Alberto TONDA

Giulio FERRERO

Pietro BARBIERO

Candidate

Alessandro BALDAZZI

June 2023

Summary

Studying bacteria and viruses is crucial for improving cures and preventing large-scale epidemics or pandemics. Antimicrobial resistance (AMR) refers to the ability of microorganisms to withstand the effects of antimicrobial treatments and poses a significant public health threat, so understanding its prevalence, mechanisms, and spread is a global priority. Research in this sector brought significant advancements in technology, which have resulted in remarkable cost reduction and accelerated processing speeds.

Despite these improvements, the acquisition of such technologies can still pose a considerable financial burden in low-resource settings. There remains ample room for further enhancements, particularly through the integration of machine learning and computational intelligence strategies to automate processes.

In the past, algorithmic solutions to bacterial resistance were not feasible due to the complexity of microbial genomes, limited data availability, and lack of computational power. Although classical algorithms are still impractical for such complex problems, the increased data and computational power have enabled initial attempts. Machine learning algorithms have been extensively studied for predicting antimicrobial resistance based on genotype information. These approaches can be supervised or unsupervised. Some studies have used gene presence/absence and antimicrobial susceptibility testing outcomes as features to train models. Recent studies have utilized k-mers derived from the genomes of antimicrobial-resistant and susceptible species, along with susceptibility testing outcomes, to develop prediction models. Tools like Mykrobe predictor and RAST employ machine learning classifiers to identify antimicrobial resistance genes. These tools have shown high accuracy in identifying resistance elements in independent validation sets. However, machine learning classifiers depend on training data and existing knowledge, necessitating large datasets of curated resistance genes and accurate genotypic data. While machine learning shows promise in antimicrobial resistance prediction, it still has a long way to go before it can be used for rapid diagnostics and replace traditional culture techniques and susceptibility testing, which typically take days or weeks to produce results.

On the other hand, no attempts involving evolutionary computation have been

attempted, despite it offering several advantages compared to machine learning approaches. Firstly, it excels at exploring vast solution spaces, making it effective in complex problems with high-dimensional search spaces. It can handle non-differentiable and discrete problems, making it suitable for tasks involving categorical variables or combinatorial optimization. Additionally, evolutionary algorithms are robust to noisy and incomplete data, producing meaningful results even when data quality is imperfect. They are designed for global optimization, avoiding getting trapped in local optima. Evolutionary algorithms are adaptable to dynamic environments, adjusting to changing conditions. Moreover, they can be parallelized for faster convergence and scalability, making them suitable for complex real-world scenarios.

For these reasons the primary objective of this research is to utilize Evolutionary Computation to address the complex issue of bacterial resistance. Specifically, the algorithm aims to identify distinct markers that are highly prevalent in resistant variants but largely absent in susceptible ones. These markers have the potential to be correlated with specific types of resistance, offering valuable insights into the underlying mechanisms of bacterial resistance. Moreover, this research endeavors to contribute to the development of targeted interventions by analyzing and uncovering the significance of these markers. A critical factor influencing the final outcome is the ability to identify a wide range of well-established markers associated with a particular bacterium and a specific antibiotic. This focus on discovering numerous known markers is vital for improving the accuracy and reliability of future research in this important area.

Three main tests have been designed to evaluate the effectiveness and reliability of the algorithm, assessing overall performance, result variability and ability to avoid under/overfitting. The initial test aimed to assess the algorithm's performance with a random seeding. This allowed for an evaluation of the algorithm's ability to navigate away from local optima and determine the impact of unfiltered data before implementing significant removals. By exposing the algorithm to unfiltered data, valuable information and crucial patterns could be identified, avoiding premature discarding of data. This test established a baseline understanding of the algorithm's behavior and guided subsequent analyses. The second test employed a targeted seeding approach, selecting only sequences from variants resistant to the specific antibiotic of interest. This strategy aimed to increase the likelihood of discovering meaningful genetic sequences associated with resistance and expedite convergence towards favorable solutions. The targeted approach was expected to yield a higher average fitness score and greater variance in results compared to the random seeding test. It offered insights into the diversity and complexity of the problem domain, while mitigating the risk of overfitting. The final test focused on minimizing the occurrence of repeating sequences in the final generation. The motivation behind this test was to strike a balance between fitness and diversity,

recognizing the importance of both factors in achieving comprehensive classification outcomes. The insights gained from the second test influenced this approach, aiming to ensure a rich and varied set of sequences that encompassed a wide range of resistance profiles. Emphasizing both high fitness and diversity was deemed crucial for effective classification.

In the conducted experiments, *Staphylococcus Aureus* was selected as the bacteria of interest, and *Gentamicin* was the primary antibiotic under investigation. The dataset consisted of 470 variants, which remained consistent across all attempts. It was expected that including more variants would improve the accuracy of the results. The population size was set to 100 individuals, striking a balance between convergence speed and performance. The iteration limit was defined as 1000 iterations, although it was not reached in any of the experiments. The data obtained from the experiments included fitness results, the number of generations needed to reach stability, and the variance of the solutions found. In Test 1, the average fitness consistently increased with each generation, peaking at generation 8. However, the final population only consisted of two distinct sequences, one being a mutation of the other with minor differences. This behavior was attributed to the vast search space and the presence of susceptible individuals complicating the identification of randomly occurring resistant sequences. The selection and replacement process favored the emergence of the "optimal" solution in terms of fitness, reducing the variability in subsequent populations. In Test 2, the introduction of selective seeding, focusing on resistant variants, resulted in improvements compared to random seeding. Stability was achieved in only four generations, and both the maximum fitness and variability increased. The maximum fitness reached the highest possible value allowed by the data, indicating significant progress. A substantial proportion of the final generation possessed the maximum fitness, suggesting the algorithm's enhanced efficiency with selective seeding. However, the number of unique individuals remained low, indicating limited variability. To address the issue of limited variability, Test 3 introduced a "re-seeding" process before evaluation. This process ensured uniqueness by replacing repeated sequences with new ones selected similarly to the initial seeding phase. This artificial increase in variability to 100% of the population made the algorithm's behavior more unpredictable, resulting in a decrease in performance and average fitness. Nevertheless, the average fitness showed a gradual increase over time, albeit with less consistent growth due to the randomness introduced by reseeded individuals. Notably, there was a notable increase in sequences with high fitness, with approximately 37% of individuals exceeding a fitness level of 20 in the 10th generation, nearly double the outcome of experiment 2.

In conclusion, the project has demonstrated that the evolutionary computation approach offers a viable solution to the problem under investigation. However, it is evident that further improvements are required to enhance the performance of the

algorithm and deliver more precise and valuable solutions.

Table of Contents

Chapter 1

Background

1.1 Bacterial Resistances Recognition

In the present times, the significance of studying bacteria and viruses has become increasingly apparent as we strive to enhance modern treatments and prevent large-scale epidemics or pandemics. However, the rapid mutation rate exhibited by these organisms poses a formidable challenge for long-term cures and effective solutions. To combat bacterial infections, a wide range of antimicrobials has been developed and discovered. Since their initial commercial introduction in the 1930s, they have proven to be invaluable tools in managing pathogenic microbes. Nevertheless, their remarkable efficacy in eliminating pathogens has also contributed to the emergence of antimicrobial resistance. Antimicrobial resistance (AMR) refers to the ability of microorganisms to withstand the effects of antimicrobial treatments. The excessive or inappropriate use of antibiotics has been closely linked to the rise and dissemination of microorganisms that are resistant to these drugs. Antimicrobial resistance is a major public threat and monitoring and understanding the prevalence, mechanisms and spread of antimicrobial resistance are priorities for global infection control strategies.
The current techniques for resistance profiling [1] follow this scheme:

1. Sample collection: DNA is extracted from colonies of different (culture-independent) or specific (culture-dependant) bacteria.

2. Sequencing: the bacterial genome is analyzed by a sequencer, like the Illumina machines, which determines the exact order of the 4 nucleotids and filters the reads by quality. The raw output of this sequencing process are what are called "Sequencing Reads", strings of A, C, T, G representing nucleotide sequences of the input DNA or RNA.

3. Sequencing-based resistance discovery: it can be of two types, both equally

important and with different pros and cons

- Assembly-based: the sequencing reads are initially assembled into contiguous fragments known as "contigs". These contigs are then compared to custom or public reference databases for annotation, enabling the identification of resistance determinants.
- Read-based: resistance determinants are predicted by directly mapping the sequencing reads to a reference database to obtain a coverage estimation and to detect variants.

4. Downstream analysis: the obtained data is rearranged for use. Some examples are

   - Resistome cluster analysis
   - Antibiotic resistance gene network analysis
   - Resistance profiling

More specifically, the Illumina machines follow the NGS (Next Generation Sequencing) [2], which is a series of steps used by modern sequencing machines. The main steps, excluded sample collection and data analysis, are:

- Library preparation: it involves preparing the DNA or RNA samples for processing and sequencing on NGS platforms. This process includes fragmenting the samples to obtain appropriately sized targets and attaching specialized adapters at both ends. These adapters facilitate the interaction of the samples with the NGS platform. The prepared samples, known as libraries, comprise a collection of molecules ready for sequencing. The specific library preparation procedure may vary depending on the reagents and methods used, but it aims to generate DNA fragments of desired lengths with adapters at both ends.

- Clonal Amplification and Sequencing: clonal amplification is a crucial step where the DNA fragments from the prepared libraries are amplified and attached to surfaces such as beads, flow cells, or ion surfaces. This amplification enables the generation of strong fluorescent signals that can be detected by the sequencers. Following clonal amplification, the library is loaded onto the sequencer, which reads and detects the nucleotides one by one through a process known as sequencing by synthesis (SBS). The SBS is a DNA sequencing technique that relies on DNA polymerases and dNTPs to replicate the target DNA strand. This method involves introducing nucleotides either individually or modified with identifying tags, such as fluorophores, to enable the recognition of the incorporated base type as the DNA molecule elongates. Each nucleotide incorporation is chemically blocked, ensuring that it is a unique and

distinguishable event. During the sequencing process, the signal produced from each nucleotide incorporation is captured. Subsequently, the blocking group is removed to prepare the strand for the next round of nucleotide incorporation by DNA polymerase. This iterative series of steps is repeated for a specific number of cycles, leading to the replication and sequencing of the desired DNA strand.

In recent years, significant advancements in technology have resulted in remarkable cost reduction and accelerated processing speeds. However, despite these improvements, the acquisition of such technologies can still pose a considerable financial burden in low-resource settings. Nevertheless, within this context, the significance of digital analysis and data collection is already widely recognized. Yet, there remains ample room for further enhancements, particularly through the integration of machine learning and computational intelligence strategies to automate processes.

Until some years ago, algorithmic solutions to the bacterial resistance problem were not feasible for many reasons:

- The complexity of bacteria's genome and his sudden mutation prevented the use of classical rule-based algorithms.
- The amount of available data was much less than today.
- There wasn't enough computational power to allow the use of AI (Artificial Intelligence) or EC (Evolutionary Computation) solutions.

Today the situation is very different: even if a classical algorithm is still unfeasible for such complex problem, the amount of data and computational power has risen enough to make initial attempts possible.

Numerous studies have investigated the use of machine learning algorithms to study antimicrobial resistance and its ability to predict resistance based on genotype information. Machine learning approaches can be implemented as either supervised or unsupervised learning methods. Some studies have used gene presence or absence and antimicrobial susceptibility testing (AST) outcomes as features to create the training dataset for models. For example, one study employed logistic regression to develop a model that could differentiate between vancomycin-susceptible and vancomycin-intermediate Staphylococcus aureus based on 14 gene parameters and 3 molecular typing markers [3]. The model achieved an 84% classification accuracy using publicly available genomic data and patient isolates, demonstrating a proof of concept for identifying antimicrobial resistance. Another study compared a rules-based approach with a machine learning-based approach (logistic regression)

for predicting antimicrobial resistance profiles and found that the machine learning-based approach had higher accuracy, especially for novel variants of known resistance genes [4].

Recent studies and tools have utilized k-mers derived from whole genomes of antimicrobial-resistant and susceptible species, along with AST outcomes, to develop prediction models. For instance, Mykrobe predictor [5] is a fast k-mer screening tool that identifies antimicrobial resistance genes and single nucleotide polymorphisms (SNPs) in S. aureus and M. tuberculosis. It utilizes curated genetic information of resistant and susceptible alleles to build reference graphs and maps k-mers from sequencing reads to these graphs. Mykrobe predictor demonstrated high sensitivity and specificity for identifying resistance elements in independent validation sets.

In contrast, Rapid Annotation using Subsystem Technology (RAST) [6] is a k-mer-based tool that employs a machine learning classifier (AdaBoost) based on the PATRIC database to identify target-specific antimicrobial resistance genes. RAST is trained on k-mer data derived from the contigs of each genome, and the binary matrix of k-mer presence/absence, along with AST outcomes, is used to form a classifier model and identify putative resistance-associated k-mers. RAST showed high accuracies in identifying specific resistance types in different pathogens.

One limitation of machine learning classifiers is their dependence on training data and existing knowledge. To apply machine learning classifiers in clinical diagnostics, a large dataset of curated antimicrobial resistance genes linked to accurate genotypic data and AST information is required to build effective and robust classifiers. Additionally, machine learning approaches are being employed to predict antimicrobial resistance genes in metagenomic data. For example, DeepArgs is a newly established tool that utilizes deep learning to identify resistance genes based on curated datasets and can predict resistance genes in new test data [7].

While the application of machine learning to antimicrobial resistance prediction and classification shows promise, these techniques still have a long way to go before they can be used for rapid diagnostic purposes and replace traditional culture techniques and AST, which typically take days or weeks to produce results.

1.2 Evolutionary Computation

Evolutionary algorithms are heuristic-based approaches that solve complex problems by emulating the principles of natural selection and evolution [8]. They are particularly effective in tackling problems that are computationally challenging or infeasible to solve optimally using traditional methods.

The core concept of an evolutionary algorithm (EA) revolves around four main steps: initialization, selection, evolutionary operators, and termination. These steps

mirror key aspects of natural selection, allowing for modular implementation and easy adaptation. In an EA, individuals with higher fitness are favored for survival and reproduction, while unfit individuals are gradually phased out, ensuring that only the fittest contribute to subsequent generations.

In the context of problem-solving, an EA aims to find the best combination of elements that maximizes a given fitness function. The algorithm continues until a termination condition is met, which could be a predefined number of iterations or reaching a specific fitness threshold. Although this scenario represents a common application in discrete problems, EAs can be employed in various other contexts.

Initialization involves creating an initial population of potential solutions or individuals, often generated randomly within the problem's constraints. The population should encompass a diverse range of solutions to explore multiple possibilities throughout the algorithm.

Selection is the process of evaluating the fitness of each individual in the population using a fitness function tailored to the specific problem. Designing an effective fitness function can be challenging, as it should accurately represent the problem's characteristics. The selection phase typically involves choosing the top-scoring individuals for further reproduction, but there could be variants to preserve some variability and try to avoid the risk of getting stuck in a local extrema.

Genetic operators encompass two sub-steps: crossover and mutation. After selecting the top individuals (often the top two), the algorithm creates offspring by combining their characteristics through crossover. This process involves mixing combinations of genetic material to produce valid solutions. This technique is common in Genetic algorithms, a subgroup of evolutionary algorithms, but is not mandatory for the algorithms to work. Mutation, on the other hand, introduces new genetic material by altering a small portion of the offspring, ensuring exploration beyond local extrema. The occurrence and severity of mutations are typically governed by a probability distribution.

Termination marks the end of the algorithm. Two common termination conditions are reaching a maximum run time or achieving a specific performance threshold. At this point, a final solution is selected and returned, marking the conclusion of the EA.

While EAs excel at optimizing solutions, it's important to note that they may not always find the absolute best solution but instead continually improve upon existing solutions. The computational requirements of EAs can be high due to the complexity of evaluating fitness: nonetheless, even seemingly simple evolutionary algorithms can tackle complex problems, as the complexity of the algorithm does not necessarily correlate with the problem's complexity.

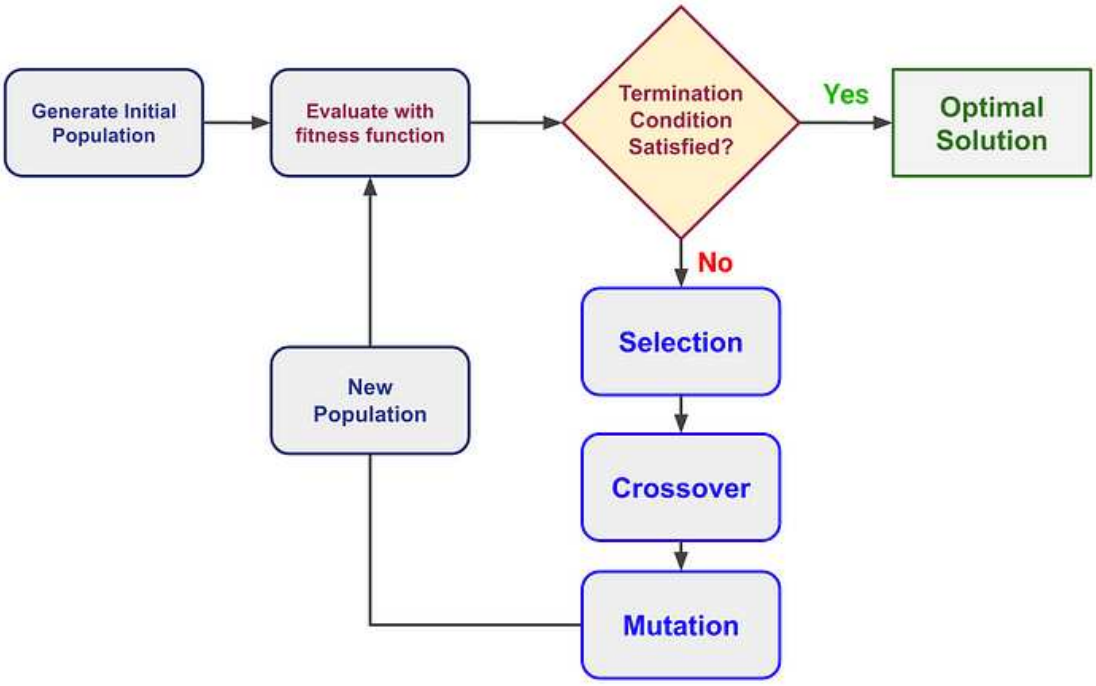

Figure 1.1: Example graph of an EA. [1]

1.3 Motivations

There are many reasons why an implementation based on Evolutionary Computation could be interesting and give additional insights in respect to the previously mentioned Machine Learning attempts.

- Exploration of Solution Space: evolutionary algorithms excel at exploring vast solution spaces, especially in complex problems with high-dimensional search spaces. They can effectively navigate through a large number of potential solutions, allowing for a more comprehensive search and the discovery of diverse and optimal solutions.
- Handling Non-Differentiable and Discrete Problems: unlike traditional optimization methods based on gradient, evolutionary algorithms can handle non-differentiable and discrete problems effectively. They are well-suited for tasks that involve categorical variables, combinatorial optimization, or problems where the fitness landscape is non-linear and discontinuous.
- Robustness to Noisy and Incomplete Data: evolutionary algorithms can handle noisy and incomplete data gracefully. By using population-based search strategies and maintaining diversity, they are less susceptible to noise

and outliers. They can still produce meaningful results even when the data is imperfect or contains missing information.

- Global Optimization: evolutionary algorithms are designed to find globally optimal or near-optimal solutions rather than getting trapped in local optima. Through mechanisms like mutation, they promote exploration of the search space, allowing for the discovery of better solutions that might be outside the vicinity of local optima.

- Adaptability and Dynamic Environments: evolutionary algorithms possess inherent adaptability and can handle dynamic environments. They can dynamically adjust the population to changing conditions, allowing them to respond and adapt to new information or evolving problem characteristics.

- Parallelization and Scalability: evolutionary algorithms can be easily parallelized, leveraging the computational power of modern hardware architectures. This parallelization enables faster convergence and scalability to larger problem sizes, making them suitable for handling complex real-world scenarios.

1.4 Theoretical and Technical Issues

The previously proposed approach encountered 3 main problems that needed to be addressed and overcome:

1. The research space is incredibly vast, posing a significant challenge in navigating and exploring it effectively

2. Extracting meaningful sequences from the complete genome without external data is nearly impossible

3. Mykrobe is a very slow tool

The first problem stemmed directly from the inherent complexity of the research domain. Each variant encompassed an extensive range of bases, typically ranging from 100 to 500 million. Consequently, selecting sequences at random from such a vast space proved to be highly inefficient, exacerbating performance issues, particularly when dealing with larger datasets. To mitigate this challenge, a solution was devised by implementing a seeder that exclusively selected random sequences from known resistant variants. This seeder strategy will be further explored and elaborated upon in paragraph 2.5, offering insights into its effectiveness and implications.
Additionally, problem 2 surfaced due to the apparently random nature of the starting and ending points of the sequences. As a result, there was a heightened

risk of selecting superfluous or incomplete sequences, which could compromise the accuracy and reliability of the analysis. To overcome this obstacle, the utilization of CONTIGS files played a pivotal role: those contains a comprehensive list of already recognized sequences recognized sequences segmented by known markers. By leveraging this resource, a more focused and precise analysis was ensured, minimizing the risk of incorporating irrelevant or incomplete genetic sequences.
Finally, problem 3, relating to the slow performance of the Mykrobe tool, which will be further explored in paragraph 3.3, was addressed to the best extent possible through the implementation of parallelization techniques. Despite these efforts, the tool remained a significant bottleneck in the overall implementation process.

Chapter 2

Proposed approach

2.1 Algorithm overview

As discussed in detail in paragraph 1.1, the field of machine learning has witnessed extensive research and produced varying outcomes. However, there has been a notable absence of applications integrating Evolutionary Algorithms and other branches of Computational Intelligence into this domain. Consequently, the primary objective of this research endeavor is to bridge this gap by applying Evolutionary Computation to tackle the challenging problem of bacterial resistance. More specifically, the algorithm aims to identify distinctive markers that exhibit a high prevalence among resistant variants while being predominantly absent in susceptible ones. These markers hold the potential for correlation with specific types of resistance. By uncovering and analyzing such markers, this research seeks to shed light on the underlying mechanisms of bacterial resistance and potentially contribute to the development of targeted interventions. In the pursuit of this goal, one of the crucial factors that significantly influences the final outcome is the ability to identify a multitude of well-established markers associated with a particular bacterium and a specific antibiotic. This emphasis on the discovery of numerous known markers is essential for enhancing the accuracy and reliability of future research in this topic.
The basic structure of this algorithm is formed by:

- A seeding phase, where the starting population is initiated
- An evaluation phase, where the fitness is calculated
- A selection and replacement phase, where the best elements from the population are selected and the worst are discarded
- A mutation phase, where each individual has a probability to receive changes to his structure

- A termination phase, where the algorithm is supposed to have reached a meaningful result, so it terminates

Algorithm 1 Basic schema of the research's algorithm

```
Population initialization                          ▷ Seeding phase
while Termination conditions do
    Selection of best elements          ▷ Selection and replacement phase
    Mutate some elements                           ▷ Mutation phase
    Calculate fitness                              ▷ Evaluation phase
end while
```

2.1.1 Initial population, seeding and selection

As explained in paragraph 1.2, the seeding is the phase in which the initial population is generated. Since the results must represent markers that could be significant to the resistance problem, the most obvious representation of individuals is made by encoding them as DNA sequences of varying length. The seeder will retrieve them from the data, check that the obtained individuals are different from one another and then encode the population adding some elements necessary for the effective implementation. The sequences are selected from the CONTIGS (see paragraph 2.5.1) and cut into random length between defined limits. Since the search space is very large, the work of the seeder is fundamental for the final success of the algorithm: this component is in fact the main element of experimentation, so its changes will be further discussed in section 3.2.
A tournament selection was used for the selection process: it involves running multiple tournaments among randomly chosen individuals and then selecting the fittest individual to continue in the following phases of the algorithm. This particular selection gives the possibility to modify the selection pressure quite easily, by simply changing the tournament size. A larger tournament size reduces the chances of weak individuals being selected since there is a higher probability of stronger individuals being present. The selection pressure parameter plays a crucial role in the convergence rate of the evolutionary algorithm, as higher selection pressure leads to faster convergence. The level of control over the selection pressure enables greater influence over the overall outcomes, resulting in improved avoidance of overfitting or underfitting.
A plus type replacement strategy was also used, which means that the entire existing population is replaced by the best population-many elements from the combined set of parents and offspring.

2.1.2 Mutators

The mutators employed in this project are specifically designed to manipulate the genomic sequence, thereby imposing certain limitations on the range of possibilities. Two primary mutators are utilized: the length mutator and the sequence mutator, each serving a distinct purpose within the algorithm.
The length mutator operates on the original sequence selected by the seeder and introduces alterations by either shortening or lengthening it. The degree of alteration is determined by a random number of basis, typically ranging between 1 and 20 in my implementation. Notably, the basis for alteration are not randomly chosen but are derived from the original sequence itself. This approach ensures that the modifications remain connected to the inherent characteristics of the sequence.
On the other hand, the sequence mutator takes the same variant from which the original sequence was derived but replaces it with an entirely different one.
In contrast to the length mutator, the sequence mutator represents a departure from the conventional evolutionary variator. It may not adhere, at least apparently, to the principle of locality as the changes introduced can be substantial. Both the length and the bases themselves can be altered, potentially leading to significant variations. However, this mutator possesses the potential to yield intriguing results, particularly when certain variants exhibit multiple sequences associated with resistance. It also presents an opportunity to unveil potential hidden correlations between specific genomic sequences.
Additionally, there was contemplation regarding the inclusion of a mutator that adds random bases to the sequences. However, the probability of generating a meaningful sequence diminishes with each additional base. Furthermore, even if a generated sequence produced using this method displayed a connection to resistance, it would be challenging to establish its significance or demonstrate its utility within the algorithm or without a broader research context.

2.1.3 Evaluation and Termination

The fitness function is quite simple and is implemented thanks to the use of a screening tool (in our case Mykrobe was used, described in paragraph 2.5.2). The screener analyzes the sequence and confronts it with a list of variants utilizing the fitness function

$$fitness = R - S$$

where R is the number of resistant correspondences and S is the number of susceptible ones. Without some more data this simple fitness function was the best option to obtain meaningful results: a variant can be to add parameters to the fitness function like

$$fitness = a * R - b * S$$

where a and b are weights added to make the presence in the resistant or susceptible variants more or less impactful on the final results. It's to be noted that these weights wouldn't affect sequences present in only resistant or only susceptible individuals, but any sequence in between.
Since the algorithm has the objective to find an high number of sequences with the highest fitness as possible, the main termination condition is

$$\max\left(fitness_n\right) - \text{avg}(fitness_n) < p$$

where $fitness_n$ is the set of fitnesses of the current population and p is a variable chosen arbitrarily which represents the maximum difference between the maximum and the average fitnesses. In this way the algorithm is forced to iterate until the population fitness is stabilized. Another condition is the number of maximum iterations, a pretty common condition in evolutionary algorithms.

Chapter 3

Experiments and Results

3.1 Tests

In order to evaluate the effectiveness and reliability of the algorithm, a series of three tests were designed. These tests aimed to assess key factors, including overall performance, result variability, and the algorithm's ability to avoid both overfitting and underfitting the problem at hand.

3.1.1 Test 1

The initial test conducted involved providing the algorithm with a random seeding, which might initially appear counterintuitive considering the challenges outlined in point 2.3. However, given that one of the fundamental advantages of an Evolutionary Algorithm (EA) compared to greedy algorithms lies in its ability to navigate away from local optima, it was essential to examine the potential impact of including a substantial amount of unfiltered data before making any significant removals. Depending on the characteristics of the research space, the results could vary from a prominent and dominant single local minimum to a complete absence of meaningful or valuable results. This test allowed for a comprehensive evaluation of the algorithm's performance under different seeding conditions, thereby informing subsequent decisions regarding data removal or refinement strategies. By intentionally exposing the algorithm to unfiltered data, the possibility to avoid prematurely discarding valuable information and potentially missing out on crucial patterns or insights is avoid as much as possible. Ultimately, this test helped establish a baseline understanding of the algorithm's behavior and paved the way for more informed and targeted analyses in subsequent iterations.

3.1.2 Test 2

The second test aimed to provide the algorithm with a non-random seeding approach, which involved selecting only sequences from variants that exhibited resistance to the specific antibiotic of interest. By utilizing this targeted seeding strategy, the expectation was to increase the likelihood of discovering meaningful genetic sequences associated with resistance. One of the anticipated outcomes of this test was to establish a starting population with a higher average fitness score compared to the random seeding test (Test 1). By deliberately choosing variants that displayed resistance to the antibiotic, the initial population would inherently possess a greater potential for containing relevant genetic markers linked to resistance. This targeted approach sought to expedite the algorithm's convergence towards more favorable solutions from the outset. Additionally, the second test was expected to yield greater variance in the final results compared to Test 1. By focusing the seeding on resistant variants, the algorithm had the opportunity to explore a wider range of genetic sequences that might be associated with resistance. This increased variation in the final results would provide valuable insights into the diversity and complexity of the problem domain, offering a broader perspective on potential solutions. Although there still remained the possibility of the algorithm overfitting to specific patterns within the data, this concern was expected to be less problematic compared to Test 1.

3.1.3 Test 3

The final test was specifically devised to compel the algorithm to minimize the occurrence of repeating sequences within the final generation. Given that this study primarily entailed a classification problem rather than pure optimization, having a greater number of sequences with high fitness was deemed more desirable and meaningful compared to a single highly fit sequence. The rationale behind this test stemmed from the insights gained from the results of the second test, which will be further discussed in paragraph 3.2.3. The motivation behind this test was to strike a balance between fitness and diversity, acknowledging the importance of both factors in achieving a more comprehensive classification outcome. While the algorithm's ability to identify highly fit sequences was crucial, it was equally important to ensure a rich and varied set of sequences that encompassed the breadth of resistance profiles.

3.2 Implementation

The effective implementation of the algorithm was developed in *Python* using the *inspyred* framework, an open-source framework for creating biologically-inspired

computational intelligence algorithms such as evolutionary computation, swarm intelligence, and immunocomputing. The *inspyred* library played a crucial role in automating the overall structure of the algorithm and implementing key components like tournament selection and the plus replacement strategy. However, certain components required customization due to their specific nature:

- The seeder, where the population is initially formed
- The mutator, to introduce random changes and variability to the population
- The evaluator, the component used to calculate the fitness

To seed a population of N individuals (for the experiments in paragraph 3.1 $N = 100$) a set of random sequences (which were only from resistant variants in experiments 2 and 3) are selected from the CONTIGS files, a subsequence is selected with random starting and ending points and then an individual is encoded. The individual is composed of:

- The DNA sequence
- Starting and ending points
- A reference to the original CONTIG file, used by the mutators
- An identifier for the original sequence

Only the DNA sequence is necessary for the theoretic algorithm to work, but the other elements are necessary implementation-wise.
The length mutator changes only the starting and ending points and update the subsequence, by a random number between 1 and 20. Checks are in place to avoid to cross the limits of the original sequence. The sequence mutator on the other hand, takes the CONTIG file of the original sequence and select randomly a different one, obviously with different starting and ending points, too.
The evaluation process is strictly dependant on Mykrobe for this implementation. Mykrobe [9] is an open-source tool that analyses the whole genome of a bacterial sample and predicts which drugs the infection is resistant to. It supports Illumina sequences and, at the moment, supports *Mycobacterium Tuberculosis*, *Staphylococcus Aureus*, *Shigella Sonnei* and *Salmonella Typhi* sequences. It does so by confronting the bacterial genome with his *panels*, which are the program correspondent of the AST files we discussed previously. An additional highly valuable feature of Mykrobe is the ability to incorporate custom panels, which allows the inclusion of information about samples that may not be available in the standard Mykrobe dataset or to focus solely on specific samples of interest. In the context of the implementation process, this particular feature of Mykrobe acquires fundamental

importance. The RAWS files, which corresponds to the utilized CONTIGS, serve as the basis for creating the custom panel. This custom panel is then utilized to determine the correspondences between the analyzed individuals and resistant or susceptible variants. By comparing the individual's genomic data with the custom panel, Mykrobe identifies the number of correspondences to resistant and susceptible variants. This count of correspondences is subsequently utilized to calculate the fitness of each individual in the algorithm. The data and statistics of the current population, including average, median, maximum, and minimum values, are saved to monitor and assess the algorithm's performance. This information will be used to generate graphs in paragraph 3.2.

Algorithm 2 Algorithm implementation

```
for N = 0, N <= 100, N = N + 1 do
    A random sequence is selected from the CONTIGS files
    A subsequence is obtained by randomizing starting and ending points
    An individual is encoded using that subsequence
    If the individual is unique, is added to the population
end for
while max (fitness_n) − avg(fitness_n) < p do
    inspyred.ec.selectors.tournament_selection
    inspyred.ec.replacers.plus_replacement
    for Each individual do
        if random number between 0 and 1 <= mutation_rate then
            if random number between 0 and 1 <= 0.5 then
                Select the current sequence from the CONTIGS
                Variate the starting and ending points
                Save that as the new sequence
            else
                Select the current variant
                Select a new sequence from the same variant
                Save that as the new sequence
            end if
        end if
        Save the new population
    end for
    for Each individual do
        Put the sequence in Mykrobe
        Mykrobe confronts the sequence with all variants to find matches
        Mykrobe returns R and S
        Calculate fitness = R − S
    end for
    Save population results to a file
end while
Show best result
```

3.3 Data and Formats

The basic data needed to train the algorithm are:

- RAWS: the preprocessed clean reads, obtained from the INSDC Sequence Read Archive, and initially sequenced by an Illumina sequencer
- CONTIGS: sequencing reads reassembled into continuous fragments obtained by an Assembly-based sequencing method. These are divided in Mapped and Unmapped, depending on the found correspondence compared to that bacterium's reference genome.
- AST: the list of known susceptibilities/resistances, the results of the downstream analysis following the traditional resistance profiling techniques.

The *fastq* format is a text-based format for storing both a biological sequence and its corresponding quality scores. It's formed of 4 line-separated fields:

- Field 1 begins with a '@' character and is followed by a sequence identifier and an optional description
- Field 2 is the raw sequence letters
- Field 3 begins with a '+' character and is optionally followed by the same sequence identifier
- Field 4 encodes the quality values for the sequence in Field 2, and must contain the same number of symbols as letters in the sequence.

In our specific case Field 1 contains an NCBI-assigned identifier and an additional description, holding the original identifier from Illumina plus the read length.
In Field 4 the byte representing quality runs from 0x21 (lowest quality; '!' in ASCII) to 0x7E (highest quality; '~' in ASCII).

3.4 Experiments

In the conducted experiments, *Staphylococcus Aureus* was chosen as the bacteria of interest, and *Gentamicin* was the primary antibiotic under investigation. The dataset consisted of 470 variants and was consistent across all attempts: it's expected that the accuracy of the results would improve with the inclusion of more variants. The population size was set to 100 individuals, as a smaller population converged too quickly, while a larger one yielded better results at the expense of performance. The iteration limit was defined as 1000 iterations, although this limit was not reached in any of the experiments.

The mutation rate was set to 33%, so around $\frac{1}{3}$ of the element of each population should receive a length or sequence mutation.
It is worth noting that in all experiments, the maximum fitness achieved was relatively low, typically ranging between 15 and 26. This outcome was not attributed to project failure or implementation errors, but rather to the dataset composition, which contained a higher number of susceptible variants compared to resistant ones. To address this issue, a larger and more diverse dataset would be beneficial. However, due to limitations in computational power, the presented results represented the best compromise between the amount of data and computational time achievable. In regard to the data obtained, we will discuss the fitness results, the amount of generation needed to reach stability and the variance of the solutions found.

3.4.1 Random Seeding

The first experiment, corresponding to Test 1, obtained the results very much compatible with what expected prior.

Generation	0	1	2	3	4	5	6	7	8
Average Fitness	-201.46	-60.18	-3.08	4.64	11.11	16.07	18.74	21.11	25.0
Maximum Fitness	26.0	16.0	18.0	18.0	22.0	24.0	25.0	25.0	25.0
Minimum Fitness	-412.0	-120.0	-11.0	-2.0	5.0	13.0	16.0	18.0	25.0
Number of different individuals	100	39	15	8	7	8	11	9	2

According to the experimental findings, the average fitness of the population demonstrates a consistent increase with each successive generation, peaking at generation 8. While this may initially appear as an optimal outcome, closer examination of the data reveals that the final population consists of only two distinct sequences. Notably, one of these sequences is a mutation of the other, with only a few bases differing between them. This behavior can likely be attributed to the vast search space involved in the study. As anticipated prior to the experiment, the presence of susceptible individuals within the research space adds complexity to the identification of randomly occurring sequences associated with antimicrobial resistance. Furthermore, the selection and replacement process implemented during the evolutionary algorithm progressively reduces the variance in the subsequent population, favoring the emergence of the "optimal" solution in terms of fitness.

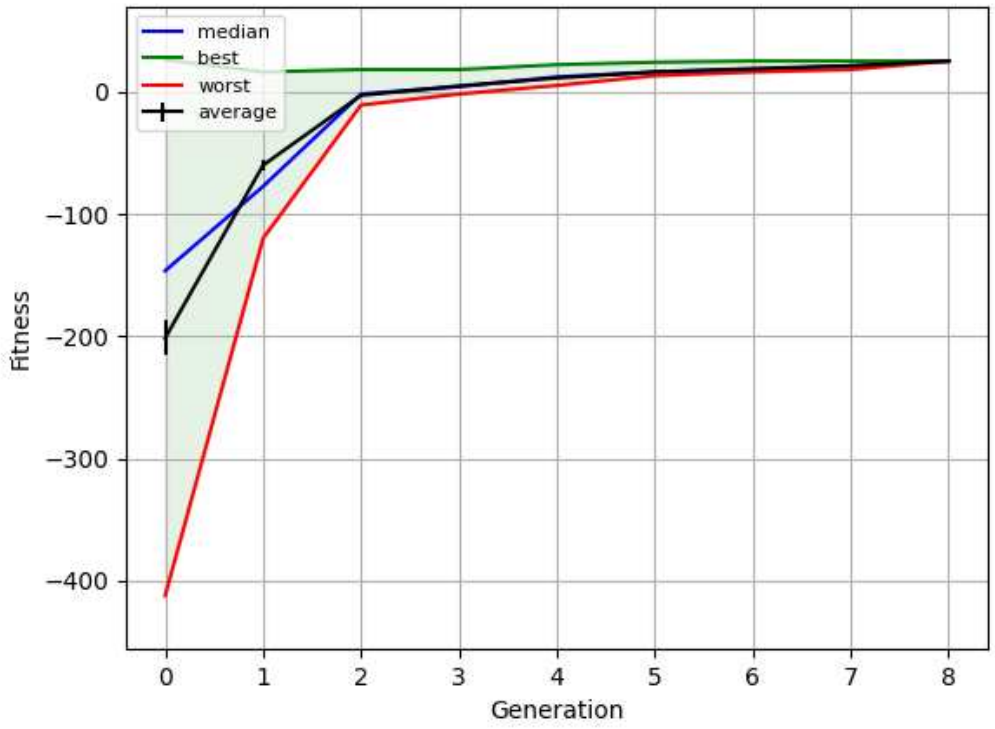

Figure 3.1: Plot of the results of Test 1 for each generation

3.4.2 Selected Seeding

The introduction of the selection of resistant variants in the seeding process showed some improvements compared to the random seeding experiment.

Generation	0	1	2	3	4
Average Fitness	-157.94	15.01	23.53	25.46	25.74
Maximum Fitness	26.0	26.0	26.0	26.0	26.0
Minimum Fitness	-412.0	-11.0	20.0	25.0	25.0
Number of different individuals	100	42	30	26	17

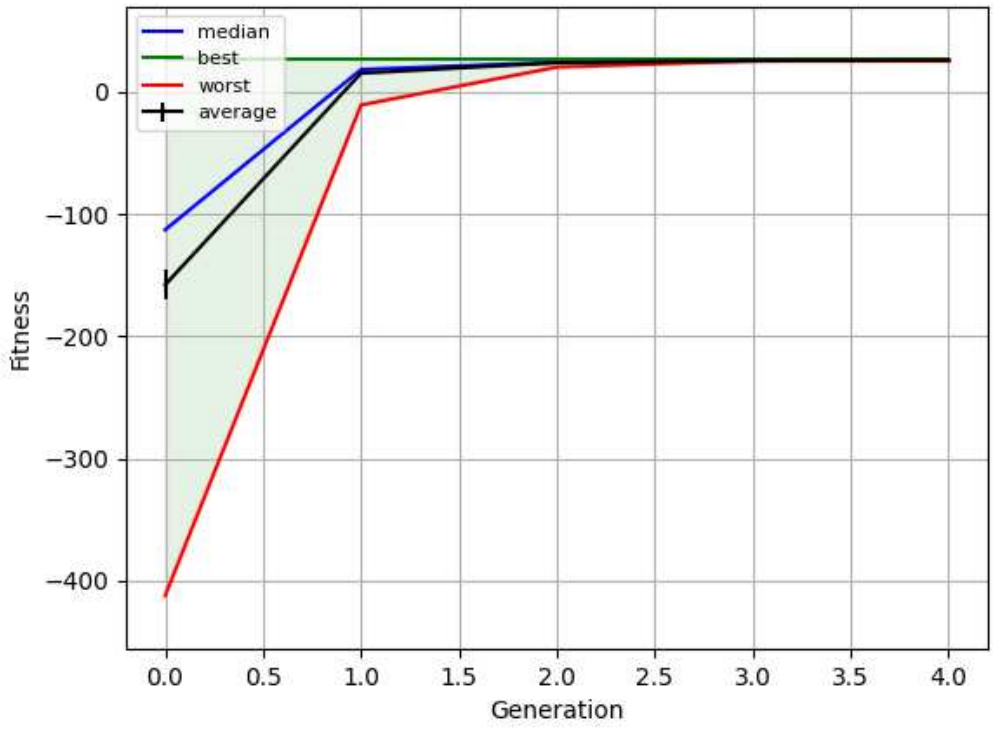

Figure 3.2: Plot of the results of Test 2 for each generation

Through the utilization of selective seeding, notable improvements have been observed in the evolutionary algorithm. In this case, stability is achieved after a reduced number of four generations, and both the maximum fitness and variability have increased compared to previous experiments. The maximum fitness now reaches the highest possible value allowed by the data, indicating a significant advancement from the earlier experiment where this maximum value was not attained, likely due to the inherent randomness in certain algorithmic phases. Furthermore, in the final generation, an impressive proportion of 12 out of 17 distinct sequences obtained the maximum fitness, accounting for approximately 70%. These encouraging results were consistently replicated across tests involving different antimicrobials, suggesting that the algorithm has become notably more efficient with the incorporation of selective seeding. However, it is important to note that despite these advancements, the number of unique individuals still only represents a mere 17% of the entire population. While this is an improvement compared to the previous experiment, it still indicates a limited degree of variability, which can be considered suboptimal.

In order to address the issue of limited variability in the population, Test 3 incorporates a "re-seeding" process. This additional step occurs before the evaluation phase, where the algorithm verifies the uniqueness of individuals and replaces any repeated sequences. The new sequences are selected using a similar approach as the initial seeding phase, effectively artificially setting the variability to 100% of the population. The results of the first 10 generation of this modified algorithm are the following:

Generation	0	1	2	3	4	5	6	7	8	9	10
Average Fitness	-139.70	-71.49	-66.87	-39.8	-24.21	-16.33	-25.76	-12.61	1.26	3.69	-4.3
Maximum Fitness	25.0	25.0	26.0	26.0	26.0	26.0	26.0	26.0	26.0	26.0	26.0
Minimum Fitness	-412.0	-143.0	-293.0	-211.0	-195.0	-301.0	-128.0	-395.0	-114.0	-113.0	-411.0

The experimental data demonstrates that this re-seeding process introduces greater unpredictability to the algorithm's behavior, resulting in a significant decrease in performance and average fitness. However, this outcome was expected since the most successful sequences are less likely to be duplicated unless they undergo mutation. Although the average fitness in Test 3 is lower compared to previous experiments, it exhibits a gradual increase over time. The growth rate, however, is less consistent due to the randomness introduced by the reseeded individuals. Despite this, the most intriguing aspect of the final result is the notable increase in sequences with high fitness. In the 10th generation, approximately 37% of the individuals possessed a fitness level exceeding 20, which is nearly double the outcome of experiment 2.

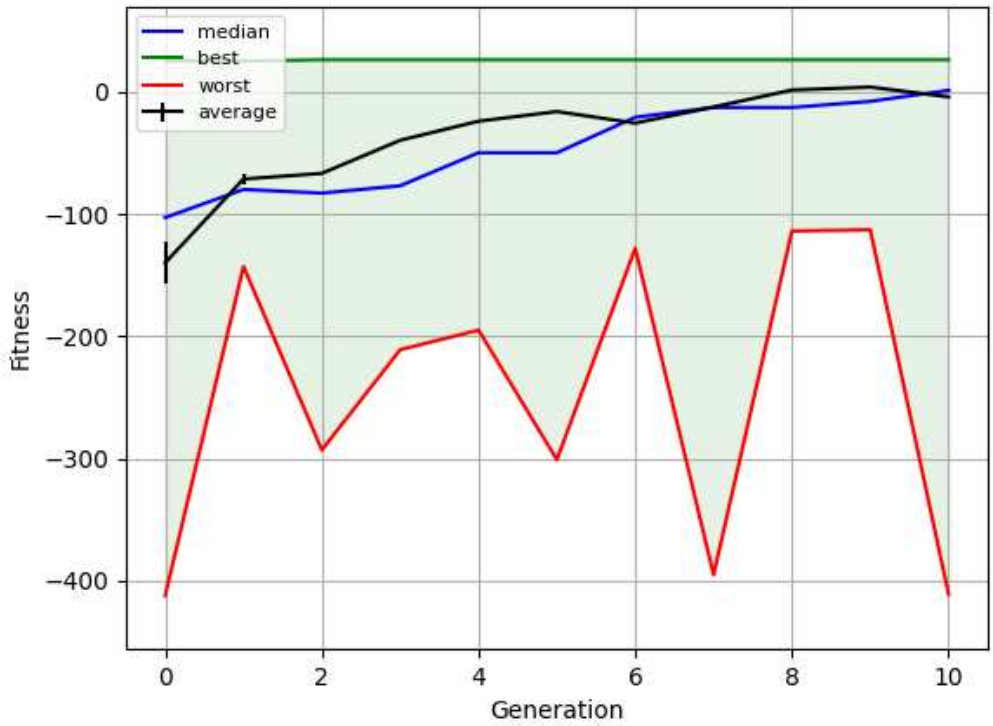

Figure 3.3: Plot of the results of Test 3 in the first 10 generations

Chapter 4

Conclusions

Evolutionary computation has proven to be a valuable tool for addressing the antimicrobial resistance problem by identifying sequences correlated to bacterial susceptibility. However, there are several caveats and issues that need to be addressed through further research and development.

The first issue encountered is the speed of the algorithm. While evolutionary algorithms have the ability to continue and mutate the results to find optimal solutions, the time required to complete a single generation can be a limitation in applications that require quick solutions. The performance of tools like Mykrobe, although essential and currently irreplaceable, contributes to this issue.

The second issue is related to the fitness and variability of the results. The algorithm tends to prioritize either fitness (as seen in experiments 1 and 2) or variance (as observed in experiment 3). However, a balance between these two aspects is crucial, as their impact on both the results and performance of the algorithm is significant.

Another noteworthy issue is that while the algorithm is efficient in finding sequences related to antimicrobial resistance, it struggles to identify those that are not prevalent in the dataset. Without parameter optimization and the development of new tools, addressing these aspects appears to be challenging.

Lastly, there is a need for a substantial amount of data. With a larger dataset, the number of sequences grows significantly, and their connection to the resistance of specific variants becomes more evident. Therefore, acquiring and utilizing a larger and more diverse dataset is crucial for improving the algorithm's effectiveness.

The results of the experiments and data analysis have provided valuable insights and have sparked ideas for further research and development of the algorithm.

One key idea is to integrate a learning component with the existing algorithms, forming a Learning Classifier System. In this approach, the evolutionary algorithm would serve as the discovery component, while the learning component would handle the classification task more effectively. This combination would leverage the

strengths of both approaches and potentially enhance the algorithm's performance.

Building upon this idea, another possibility is to incorporate two distinct datasets into the algorithm: one dedicated solely to seeding and another exclusively for evaluation. This approach aligns with the common trend in classification algorithms and can help prevent overfitting while generating more meaningful results.

Parallelization is another important aspect that can be explored. By optimizing the code and improving the management of the Mykrobe tool, the algorithm could process multiple instances simultaneously, significantly speeding up the workflow.

Overall, this research serves as a demonstration of the benefits that evolutionary computation techniques can offer in medical and biological research. However, it also highlights the current limitations and complexity of the original problem, prompting further exploration and refinement of the algorithm.

Bibliography

[1] Alaric W. D'Souza Manish Boolchandani and Gautam Dantas. «Sequencing-based methods and resources to study antimicrobial resistance.» In: *Nature Reviews Genetics* 20 (2019), pp. 356–370 (cit. on p. 1).

[2] *Introduction to NGS*. URL: `https://www.illumina.com/science/technology/next-generation-sequencing.html` (cit. on p. 2).

[3] Robert A. Petit III Lavanya Rishishwar et al. «Genome sequence-based discriminator for vancomycin-intermediate Staphylococcus aureus.» In: *Journal of Bacteriology* 196 (2014), pp. 940–948 (cit. on p. 3).

[4] Tahir Hussain Mitchell W. Pesesky et al. «KPC and NDM-1 genes in related Enterobacteriaceae strains and plasmids from Pakistan and the United States». In: *Emerg Infect Dis* 21 (2015), pp. 1034–1037 (cit. on p. 4).

[5] Bradley P. et al. «Rapid antibiotic-resistance predictions from genome sequence data for Staphylococcus aureus and Mycobacterium tuberculosis.» In: *Nat. Commun.* 6 (2015), p. 10063 (cit. on p. 4).

[6] J. J. Davis et al. «Antimicrobial resistance prediction in PATRIC and RAST». In: *Sci. Rep.* 6 (2016), p. 27930 (cit. on p. 4).

[7] Arango-Argoty et al. «DeepARG: a deep learning approach for predicting antibiotic resistance genes from metagenomic data.» In: *Microbiome* 6 (2018), p. 23 (cit. on p. 4).

[8] J.E. Smith and Agoston E. Eiben. *Introduction to Evolutionary Computing*. Springer, 2013 (cit. on p. 4).

[9] *Mykrobe documentation*. URL: `https://github.com/Mykrobe-tools/mykrobe/wiki` (cit. on p. 15).