Master Thesis

Experiments in In Silico Evolution with Aevol

Brian Davis

Supervisor: Berenice Batut

April 2020

Albert-Ludwigs Universität Freiburg

Bioinformatics Group

Professor Dr. Rolf Backofen

Abstract

Not all aspects of evolution are fully understood, and one area of active interest is reductive evolution, in which the genome of an organism evolves to become significantly smaller over time. Among the more well-known examples of this phenomena is *Prochlorococcus*, a marine cyanobacteria whose genome is up to 50% smaller than its closest living relative, *Synechococcus*. To study the mechanisms behind reductive evolution in the lab would be too costly, expensive, and slow, so we turn instead to *in silico* evolution. This method seeks to simulate organisms and their evolution in software, allowing for greater control of the environment, mutation rates, and other variables, as well as providing a full record of all organisms in a lineage. In this thesis we use the in silico tool *Aevol* to study reductive evolution, particularly by looking at how varying parameters (e.g. mutation and selection rates, population size) impacts the structure of the genome as well as measures such as robustness, evolvability, and fitness. Through these methods we hope to shed some light on the underlying mechanisms which lead to reductive evolution.

Contents

Abstract **I**

List of Figures **III**

List of Tables **V**

1 Introduction **1**
1.1 Problem Statement 2
1.2 Report outline 3

2 Background **4**
2.1 Experimental Evolution 4
2.2 Aevol 6
2.2.1 Aevol's Architecture 6
2.3 Analyzing with Aevol 11
2.3.1 Evolvability 12
2.3.2 Robustness and Antirobustness 12
2.3.3 Fitness 13
2.4 Related Work 13

3 Methods **15**
3.1 Contributions 15
3.2 Experimental Designs 15
3.2.1 Inputs 16
3.2.2 Evaluation Strategy 17

3.2.3 Structure 20
3.3 Expected Results 21

4 Results and Discussion **22**
4.1 Results 22
4.1.1 Statistical Analysis 22
4.1.2 Genome Size 22
4.1.3 Metabolic Error 24
4.1.4 Genome Structure 24
4.1.5 Evolvability 27
4.1.6 Robustness 29
4.2 Discussion 29
4.2.1 Relation to Real-World Biological Entities 29
4.2.2 Limitations of Results 29

5 Conclusion and Future Work **31**
5.1 Conclusion 31
5.2 Future work 31

List of Figures

1.1 Unknown phylogeny 2

2.1 Overview of Aevol's architecture. 7
2.2 Overview of Aevol's concept of fitness. 10
2.3 Lineage basic illustration. 12

4.1 Genome size 23
4.2 Metabolic error 24
4.3 Non-coding DNA 25
4.4 Mean number of functional genes 26
4.5 Average size of functional genes 27
4.6 Evolvability boxplot 28
4.7 Robustness bar graph 29

List of Tables

3.1 Table of parameters 16
3.2 Aevol robustness statistics 19
3.3 Aevol's stats: genes and base pairs 20
3.4 Aevol's stats: fitness and mutation 21

4.1 Evolvability mean and standard deviation 28

Chapter 1

Introduction

Reductive evolution is the process of the genome of an organism shrinking over time, with respect to both the number of base pairs and genes. Some species of bacteria have experienced reductive evolution over the course of millions of years, and this reduction in their genome has lead to a loss of genes, certain regulatory abilities, etc. For example, some strains of the marine cyanobacteria *Prochlorococcus* have experienced a reduction of nearly 40% of their base pairs when compared to larger strains of their closest living relative, *Synechococcus*[15]. Despite being extensively studied, the mechanisms and full impact of reductive evolution are not fully understood and are an area of ongoing research.

Although it would provide more conclusive evidence, performing *in vivo* experiments is often impractical because of the difficulty or impossibility of reproducing natural environmental conditions in a lab. Such experiments are often too costly in terms of both time and resources. As an alternative, *in silico experimental evolution* is one option that can be used to study the conditions under which an organism's genome may become reduced. In this method, organisms and their evolution over thousands or millions of generations are simulated in software. In this manner, one can control and evaluate every aspect of their evolution over time and a full record of their lineage may be maintained and studied, allowing one to go back and closely examine every step of the evolutionary period for a greater understanding of the factors that lead to specific effects on the genome. The in silico tool *Aevol* is one such platform which realistically models bacterial genomes and

evolution, allowing one to draw conclusions about their real-world counterparts. In the following thesis, we present the results of our experiments in artificial evolution which aim to identify and evaluate several factors which potentially lead to changes in genome structure and a reduced genome in simulated bacteria using the Aevol platform.

1.1 Problem Statement

Among the difficulties of studying reductive evolution with in vivo evolutionary experiments, one of the most difficult obstacles to overcome is the lack of a full ancestral record. This lack of a full phylogeny can make it difficult or impossible to tell exactly when and how a specific event occurred, or a trait evolved or was lost, as illustrated in Figure 1.1 below. In this exam-

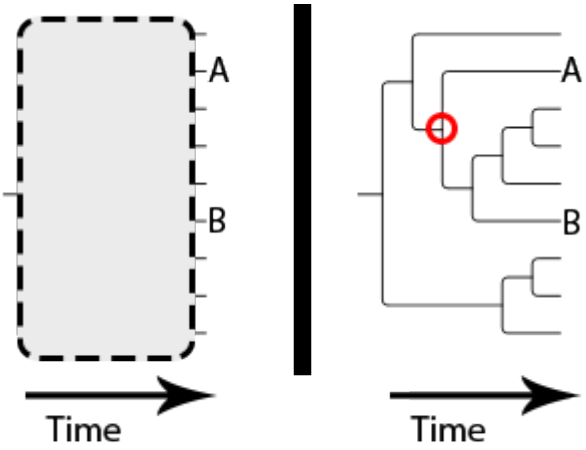

Figure 1.1: An illustration of unknown phylogeny. Since the phylogenetic information under the shaded box is typically not known, the point of divergence (red circle) can't be determined.

ple, we are comparing two related organisms A and B and we are trying to determine when and how a specific trait was gained or lost by one of the organisms. This may be useful, for example, if we are attempting to estimate the relative importance (due to conservation over many generations) of some trait. Without the phylogenetic information (under the shaded box) we may not be able to identify the point in their evolutionary history at which the two organisms diverged, making time estimates difficult or impossible.

Another major downside to *in vivo* evolutionary experiments is that they are slow. For example, the well-known E. coli Long-Term Evolution Experiment (LTEE) by Profesor Lenski at Michigan State University has been ongoing since February of 1988 and only passed generation 65,000 in 2016, 28 years later.

As an alternative to in vivo experiments, *in silico* evolutionary experiments are well-suited to the task of studying reductive evolution. Generations of organisms may be evolved within a very short time period, and a full "fossil record" of each lineage may be kept on disk for further analysis.

The in silico tool Aevol has a realistic artificial chemistry model which was developed specifically to study genome structure. It contains tools to analyze the robustness, fitness, and evolvability of digital organisms over time.

1.2 Report outline

This chapter serves as the introduction to the thesis and the research problem we are facing. In Chapter 2, we provide some necessary background information on reductive evolution, in silico evolution in general, and Aevol in particular. Chapter 3 describes our experimental setup. Chapter 4 provides the results and analysis of our experiments, and Chapter 5 presents our conclusions.

Chapter 2

Background

In this chapter we will examine some of the theoretical background information required for this thesis. We begin with an examination of experimental evolution and move on to a discussion of Aevol, the specific tool that was used in this thesis. We close the chapter by examining how Aevol can be used to study reductive evolution and discuss the current state of the literature surrounding reductive evolution.

2.1 Experimental Evolution

As discussed in Section 1.1, in vivo experiments, although sometimes more realistic, have their own set of difficulties. Some examples of these difficulties include recreating challenging environmental conditions (e.g. simulating the open ocean in a lab) and identifying and/or simulating the multiple selection pressures acting on genomes in the real world [1]. These challenges add enormous difficulty and complexity to conducting proper experiments and isolating the specific factors which lead to particular outcomes.

In silico evolution simulates organisms in software, thus allowing for far greater control and analysis of the environment and other experimental conditions. In contrast to in vivo experiments, a greater amount of control is also available with regard to the way organisms may interact, reproduce, and evolve. For example, a genome may be created completely from scratch or an existing genome may be fed into the simulation. Reproduction rates can depend on overall fitness, on relative fitness, or some other criterion.

Factors such as the mutation rates or selection pressure are then parameters for the model and may be kept constant or allowed to vary over time. Given that these are parameters of the system, they may be tightly controlled, leading to a clearer picture of the factors influencing different outcomes. An underlying deterministic model can also allow for a reconstruction of the system from any given point, allowing one to easily create a record of events, including phylogenetic trees.

Why use Aevol

Many silico evolution tools have been used to test various aspects of evolution: Tierra and Avida, in which the genetic units are computer programs fighting for CPU time [12] and [11] were some of the first; DOSE, an ecology-conscious method of checking for heterozygosity (variation within a population) [3] is a more recent example; and many more (see [10] for a more full review).

The in silico tool *Aevol* was developed to "study the evolution of the size and organization of bacterial genomes in various scenarios"[1]. The program has been expanded upon and tested in a variety of scenarios over the years. Examples of such experiments include: testing the predictability of evolution with high mutation rates as in viruses [2], determining whether selection is able to overcome evolution's drive towards more complex organisms [8], examining the role of mutators in reorganizing the genome in order to overcome mutational load [13], examining the effects of population shape on levels of cooperation [9], modeling regulatory networks [14] and more.

As an in silico experimental evolution tool, Aevol embodies several of the advantages of in silico evolution in general. There is a "fossil record" of each generation, experimental conditions are tightly controlled, and experiments are easily repeatable. The encoding/decoding strategy of Aevol follows a biologically realistic model, in the sense that there are many degrees of freedom between an organism's genome and its proteome. Many genes may encode for very simple proteins (e.g. if the genes contain the same or similar sequences), and by contrast, overlapping genes may code for complex proteomes.

In the following sections, the in silico experimental evolution tool Aevol will be examined in greater detail.

2.2 Aevol

Aevol follows a "sequence-of-nucleotides" model [1] in which organisms are simulated with a binary genome which can either be generated at random or input as a previously-generated sequence. Aevol essentially consists of three steps: 1) decoding the genome of these organisms to produce artificial proteins, 2) selecting the most fit individuals and 3) reproduction of these fittest individuals with possible variations (mutations, rearrangements, etc.). In the sections below we will examine each of these steps in greater detail.

2.2.1 Aevol's Architecture

Aevol's three steps—decoding the genome, selection, and reproduction—are illustrated in Figure 2.1.

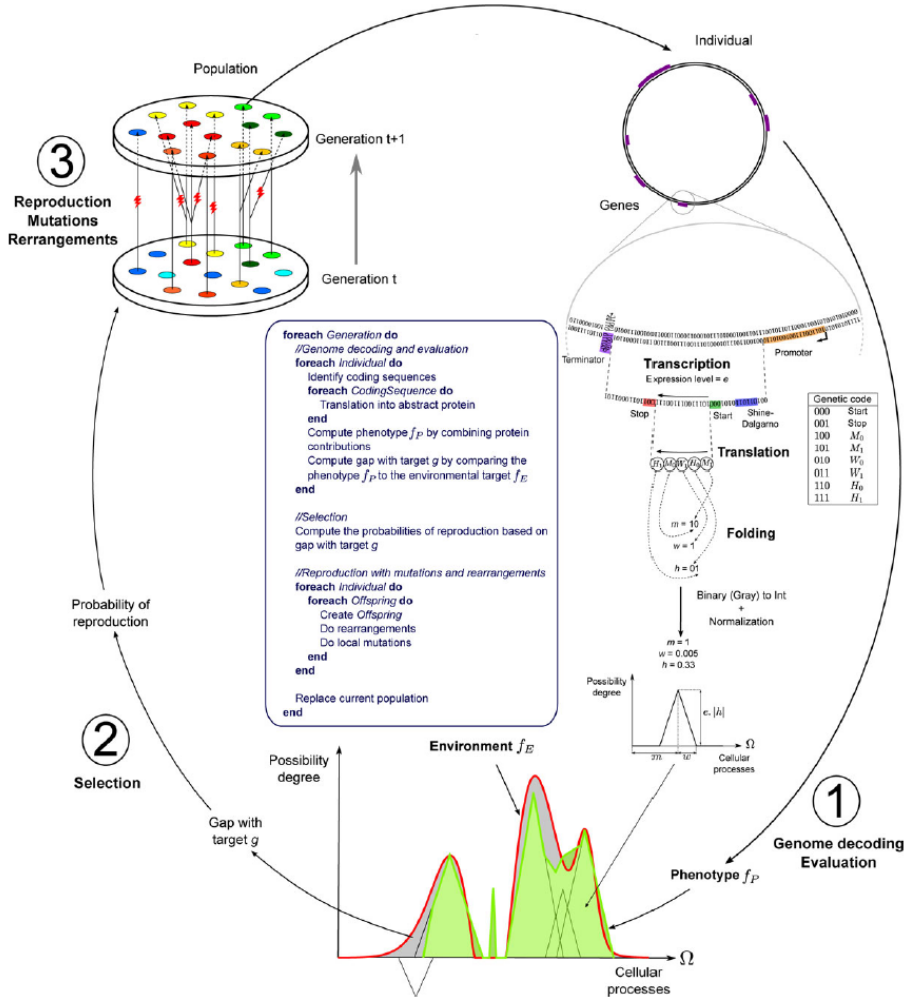

Figure 2.1: Overview of Aevol's architecture, from [1]

Decoding the Genome

In Aevol, a genome consists of a string of binary characters where 0 is complementary to 1. Each organism in the (initially clonal) population has a double-stranded circular genome which is either generated randomly or which was provided as input. To decode the genome and produce the phenotype, the sequence is searched for transcribed regions. Transcribed regions are denoted by promoter and terminator sequences. The promoter sequence is a sequence whose Hamming distance d is within $d_{max} = 4$ mismatches of the predefined consensus sequence 01010110011110010010110. Termina-

tors are sequences which can form a stem-loop structure with a stem size of 4 bases and a loop length of 3 bases (i.e. $abcd{***}\overline{dcba}$ where a is complementary to $\overline{a}$, b is complementary to $\overline{b}$, etc.). Lastly, the initiation and termination signals are sought, which are simply Shine-Dalgarno-like signals (i.e. $011011 * * * *000$ to start and $011011 * * * *001$ to stop). Lastly, an expression level e is assigned to each coding region, following the formula $e = 1 - \frac{d}{d_{max}+1}$ where d is again the Hamming distance between the coding region and the consensus sequence given above and d_{max} is the maximum allowable distance (i.e. 4).

Once an initiation sequence is found, the following bases are read three at a time (codon by codon) until a stop codon (by default 001) is found. If a stop codon is not found, then no protein is produced. Since a transcribed region may have multiple initiation signals, operons are therefore allowed. The codons following an initiation signal encode for three parameters according to the genetic code given in Figure 2.1: m (mean), w (half-width), and h (height), which together define a triangle representing a "cellular process".

A cellular process is simply an abstract representation of some phenotypic function and is represented by the ordered set $\Omega = [a, b] \subset \mathbb{R}$. Together, these cellular processes make up the organism's proteome. To keep things simple, Ω is a one-dimensional space in the interval $[0, 1]$, i.e. a "cellular process" is simply a real number, and the genomic encoding of each cellular process determines the function $f(x) : \Omega \to [0, 1]$. The mean m gives us the specific cellular process in the range $[0, 1]$. The width w describes the "scope" of the process, i.e. the *pleiotropy* of the process, meaning the subset of the protein that is in the interval $(m - w, m + w) \subset \Omega$. The height determines the degree of possibility of the process, i.e. its relative strength.

The codons are read one after the other and their Gray codes[1] are used to compute the real numbers m, w, and h as follows. Each parameter (m, w, h) is assigned two codons in the genetic code (see Figure 2.1), for example $w_0 = 010$ and $w_1 = 011$. Any w_0 codons become a 0 in the Gray code, and vice versa with 1s. So if, for example, when reading the coding sequence, the codons w_1, w_0, w_1, w_0 are read, the Gray code becomes 1010, which is 12 in decimal. This is done for m, w, and h, and the resulting values

[1] A binary encoding such that two successive values (e.g. 2, 3) only differ by at most one bit (e.g. 0011, 0010). See `https://en.wikipedia.org/wiki/Gray_code`

are then normalized to be in the proper range. w's range is specified in the parameter file (as `MAX_TRIANGLE_WIDTH`), h must be in the range $[-1, +1]$ (indicating that both activating and inhibiting processes are allowed) and m must be in the range $[0, 1]$ (the range of Ω).

Given the fact that there are likely multiple coding sequences in a genome, several triangles (cellular processes) are translated from the genome, each parameterized by its own m, w, and h. These triangles form the phenotypic function f_P. *Fuzzy logic* is used to find the overall contribution of each cellular process, using the Lukasiewicz fuzzy operators[2]. Roughly speaking, the activating proteins are added up, as are the inhibiting proteins, and the difference between these two totals represents the final function f_P. More formally, if f_i is the possibility distribution of the i-th activator protein and f_j is the possibility distribution of the j-th inhibitor protein, then the phenotype of the individual is defined as:

$$f_P = max\left(min\left(\sum_i f_i(x), 1\right) - min\left(\sum_j f_j(x), 1\right), 0\right)$$

Selection

After the genome is decoded, the organisms are tested for their fitness. Fitness in Aevol is related to the gap between the phenotype of a sequence f_P and the environmental target function f_E, as illustrated in Figure 2.1. This environmental target function f_E is a user-defined set of Gaussians which are specified in a parameter file, with each Gaussian being identified by three parameters: its height, its location along the axis, and its width. The difference between the phenotype (as calculated above) and the environmental function is the "metabolic error", labeled g in the figure and is more formally defined as: $g = \int_a^b f_E(x) - f_P(x)dx$. The idea is illustrated in Figure 2.2

Aevol contains several selection schemes but here we will only consider the `fitness_proportionate` scheme, since this was the only selection scheme employed in our experiments. In this scheme, the probability of

[2]See `https://en.wikipedia.org/wiki/Lukasiewicz_logic` for an introduction.

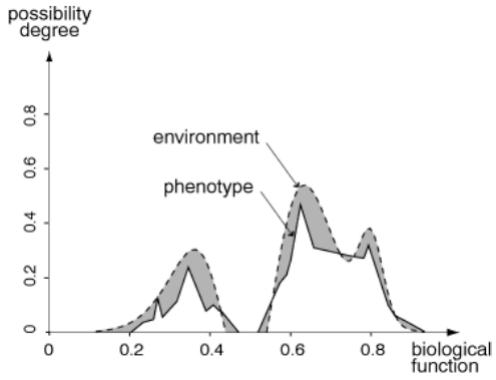

Figure 2.2: Overview of Aevol's conception of fitness, from [5]

reproduction for each organism i is proportionate to its fitness, namely:

$$P(\text{reproduction}) = \frac{e^{-k*g}}{\sum_{i=1}^{N} e^{-k*g_i}}$$

where k is a user-definable parameter which determines the selection intensity and g is the metabolic error.

Reproduction

Once the fittest organisms in the population are found and their probabilities of reproducing are calculated as described in the previous Section, new organisms are produced. This is done for each potential parent organism by drawing from a multinomial distribution with the probability of reproduction given above. The population size N is kept constant and a record of each generation is kept so that the phylogenetic lineages can be recreated. Since the population size is held constant, this implies that a single organism with a high probability of reproduction may produce multiple offspring and an organism with low probability of reproduction may produce none.

When new organisms are created and their genomes are copied from their parent organisms, it is at this stage that some of the driving forces in evolution occur, namely the possibility for variation through mutation, indels, and frameshifts. Offspring will receive their parents' genome but their genome may be subject to perturbations due to stochastic effects. Mutation rates are set in the parameter file and include point mutations, insertions

and deletions (indels), and rearrangements (duplication, deletions, translocations, and inversions).

The mutation, indels, rearrangement, etc. events are carried out by first determining the number μ of such events which will occur, based on the mutation rate specified in the parameter file and drawing from a binomial distribution (e.g. $B(L, \mu_{\text{point}})$ for point mutations, $B(L, \mu_{\text{large deletions}})$ for large deletions, etc. where L is the size of the genome). Then a random point (or points, in the case of e.g. rearrangement) is chosen and the event is carried out, with the order of these events shuffled randomly.

2.3 Analyzing with Aevol

Once the experiments have completed, Aevol by default produces several statistics files which include information about genome size, the percentage of coding DNA, number of genes, average metabolic error, and many other statistics. It further includes a number of post-treatments that allow one to analyze specific individuals or the population at large, including tools for determining robustness, evolvability, coalescence, and the lineages.

One of the key features of Aevol is the ability to look back in time at the "archaeological record" of previous organisms, which is stored on disk, in order to perform various analyses. This is primarily done with a myriad of post-treatments", i.e. supplementary programs. These post-treatments generally require a `lineage` file, which shows a record back in time of the line of descent for an individual. One may specify either the best-ranked individual (i.e. the fittest) or a specific individual by their unique identification number. The basic idea is illustrated below in Figure 2.3.

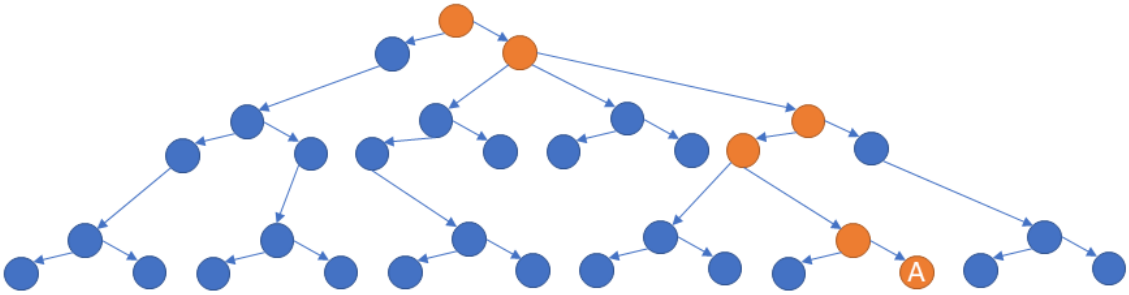

Figure 2.3: A basic illustration of a lineage. The ancestors of the individual (labeled 'A') can be traced back through the previous generations for all of its ancestors (all in orange).

In the following subsections we will examine other statistics that Aevol calculates, as these factors play a major role in reductive evolution.

2.3.1 Evolvability

Evolvability is usually defined as the ability of a system (in this case an organism) to evolve. In other words, a system has evolvability "if mutations in it can produce heritable phenotypic variation"[16]. However, of critical importance is that this is not simply having a large amount of genetic diversity, but rather *adaptive* diversity which provides some benefit.

2.3.2 Robustness and Antirobustness

Robustness is broadly defined as the ability of an organism to withstand disruptions or perturbations without affecting its phenotype. There are differing kinds of robustness, as well: one may describe robustness in terms of mutational robustness–which describes the extent to which an organism's phenotype is not affected by stochastic mutational events–or one may speak of environmental robustness, which describes the ability of an organism to maintain its phenotype in diverse environments with little or no loss in fitness.

Also important is the idea of antirobustness, wherein an organism may actually *thrive* on such perturbations. This has been suggested as a possible method of minimizing the effects of *Muller's ratchet*, wherein deleterious

mutations accumulate in a population as a result of genetic drift[4]. A large number of deleterious mutations may provide fodder for selection to fix more beneficial mutations in the population[13].

Robustness vs. Evolvability

There is, then, a seeming trade-off between robustness and evolvability. The more robust a system is, the less phenotypic variation is generated by random mutation events, and thus less evolvability. However, a key factor is to distinguish between genomic and phenotypic robustness; a strong phenotypic robustness promotes structural evolvability, as the likelihood that a mutation is deleterious is smaller in populations with more robust phenotypes. For a fuller discussion, see [16].

2.3.3 Fitness

2.4 Related Work

Much work has already been done in the field of reductive evolution, and in this section we will look at the current state of the literature.

Liard et al. showed in [8] that selection for fitness is not necessarily enough to overcome the tendency of organisms to become more and more complex. They describe the problem of a "complexity ratchet," that is, that the tendency of organisms becoming more and more complex as irreversible once the organisms had reached a certain complexity level. In their words:

> "Since gene deletion is obviously deleterious [in this scenario], the only available evolutionary path for already complex organisms is a headlong rush toward increasing complexity by acquiring new genes. Hence the ratchet clicks, further widening the fitness valley that separates the current genome from a simple one, soon making it so wide it is very unlikely to be crossed."

Echoing the findings of Knibbe et al. [6] they found, however, that limiting *robustness* can overcome the tendency of organisms to become more and more complex, because this places an upper bound on the amount of information that an organism can transmit in its genome. Increasing the mutation rate forced gene elimination despite the fitness loss because it lowered

the information content of the genome. A mutation rate of even $\mu = 10^{-4}$ resulted in nearly 40% of their organisms developing a simpler genome.

Knibbe et al. also found, via in silico experimentation, that the accumulation of non-coding DNA strongly depends on the mutation rate. This in turn affects the selection tradeoff between reliably passing on the existing genome and having the mutational variability to adapt to new challenges. Under higher mutation rates, their organisms closely resembled viral genomes in that they had almost no non-coding sequences. When the selection strength was larger, genomes were larger.

Koskiniemi et al. [7] performed in vivo experiments on the bacterium *Salmonella enterica* in which the effects on fitness of random deletions was measured. Some 25% of the deletions actually caused an increase in fitness under some conditions, suggesting that there is a certain cost associated with having superfluous genes and thus gene loss may be selected for under certain conditions.

Chapter 3

Methods

With an understanding of the basics behind us, in this chapter we provide an overview of our contributions and proposed solution.

3.1 Contributions

3.2 Experimental Designs

To assist in determining which conditions might lead to reductive evolution, we need to isolate individual variables and change just one thing at a time in order to see what effect, if any, it has on the final genome. To that end, we designed and conducted a series of experiments in which a "wild type" genome was allowed to evolve in differing conditions for 500,000 generations before analyzing the results. To first create the wild type, a genome was generated randomly in Aevol which had at least one coding gene and which was allowed to evolve for 10 million generations in a non-varying environment. By the beginning of its use in our experiments, the wild type comprised 13,237 base pairs with 132 functional genes (i.e. genes which produce a gene product). We allowed the wild types to continue to evolve in the same environment in a total of 6 different conditions: with an increased/decreased selection strength, an increased/decreased mutation rate, and an increased/decreased population size.

Additionally, a control condition was performed in which the wild type was simply allowed to continue to evolve for the 500,000 generations with

no change in any of the above conditions. To minimize bias, for each condition we performed five runs each (i.e. 5 rounds of mutation up, 5 rounds of mutation down, etc.) each with a differing random seed to control for the deterministic effects of the pseudorandom nature of Aevol's stochastic processes. This lead to a total of 35 experiments, all of which were carried out on a cluster from bwCloud[1].

The resulting data was processed using a combination of Python[2], Pandas[3], and Jupyter Notebook[4].

3.2.1 Inputs

In Table **??**, our parameter values for all input parameters may be found. Please note that for μ, this represents the mutation rates for point mutations, small insertions, and small deletions. The rearrangement rates were not changed under any condition and were always $1e-6$ for duplications, deletions, translocations, and inversions.

Condition	**Parameter**		
	μ	k	N
control	$1.00E{-}7$	1000	1024
mutation up	$4.00E{-}7$	1000	1024
mutation down	$2.50E{-}8$	1000	1024
selection up	$1.00E{-}7$	4000	1024
selection down	$1.00E{-}7$	250	1024
population up	$1.00E{-}7$	1000	4096
population down	$1.00E{-}7$	1000	256

Table 3.1: Table of input parameters. μ is the mutation rate, k is the selection strength, and N is the population size.

As can be seen in Table 3.1, only one parameter varied per condition in order to isolate potential influences. A multiplier of 4 was chosen for the differing conditions relative to the control condition (e.g. $N_{\text{population up}} = 4096 = 4 * N_{\text{control}} = 4 * 1024$). In all conditions, the environment did not

[1] `https://www.bw-cloud.org/`
[2] `https://www.python.org/`
[3] `https://pandas.pydata.org/`
[4] `https://jupyter.org/`

vary, and once the experiment was begun, the above parameters were held steady as well.

3.2.2 Evaluation Strategy

In this section, we will examine the criteria we will be using to evaluate the results. The primary criteria will be examining the evolved genome's evolvability, robustness, and structure.

Statistical Analysis of the Conditions

We must first determine how to tell whether the results of the condition (mutation up, selection down, etc.) were significantly different from the control condition, statistically speaking. To do this, we will evaluate the means of all seeds of the control condition for some test variable (e.g. robustness, evolvability, etc.) vs. the same for a different condition (e.g. mutation up) using the Wilcoxon signed-rank test. This is similar to a paired Student's t-test but is used when the distribution of the two samples cannot be assumed to be normally distributed. More precisely, we will use the "Mann-Whitney U" test, another nonparametric version of the Wilcoxon test which can be used when the sample sizes are different. The Mann-Whitney U test checks whether two independent samples were selected from populations having the same distribution. The test consists of the following steps:

1. Assign numeric ranks to all observations
2. Add up the ranks for the observations from the first sample
3. The statistic U_1 for the first sample is given by:

$$U_1 = R_1 - \frac{n_1 * (n_1 + 1)}{2}$$

   where n_1 is the sample size for the first sample and R_1 is the sum of the ranks from the first sample.

The U statistic for the second sample (i.e. U_2) is computed analogously. We used the `Scipi.stats.mannwhitneyu` function to calculate this statistic as needed, which also calculates the p-value.

If the p-value is below 0.05, we may reject the null hypothesis H_o that the two samples (i.e. the control and the condition) are from the same distribution.

Fitness

Fitness in Aevol is closely tied in to the "metabolic error". This error, g, is calculated as the gap between the environmental function f_E and the phenotype of the organism, f_P. Once g is determined as described above in Section 2.2.1, it may be used to calculate the actual fitness of the organism according to the equation:

$$\text{fitness} = exp(-k * g)$$

where k is the *selection coefficient* variable set by the user in a parameter file. Aevol provides fitness statistics both for the fittest individual and for the population at large.

Robustness

Aevol calculates statistics for both mutational robustness as well as antirobustness. Robustness in Aevol is calculated similarly to evolvability: a `lineage` file for an individual is fed to the post-treatment `misc_ancestor_robustness`, which generates a large number (specifiable by the user) of offspring, whose fitness is then measured. The percentage of these offspring which are neutral (i.e. whose phenotype is not affected by the mutations) is the robustness, and the percentage of positive offspring determines the antirobustness.

In our experiments, at the end of the run of 500,000 generations we generated a lineage file for the best individual at generation 500,000, i.e. the individual whose metabolic error was smallest. This lineage file is then fed in to the post-treatment "`aevol_misc_ancestor_robustness`" and robustness statistics are generated for each generation in the lineage file. The statistics given are summarized in Table 3.2 below.

Statistic
Fraction of positive offspring
Fraction of neutral offspring (aka reproductive robustness)
Fraction of neutral mutants (aka mutational robustness)
Fraction of negative offspring
Cumulative delta-gaps of positive offspring
Cumulative delta-gaps of negative offspring
Delta-gap for the best offspring
Delta-gap for the worst offspring
Cumulative delta-fitness of positive offspring
Cumulative delta-fitness of negative offspring
Delta-fitness for the best offspring
Delta-fitness for the worst offspring

Table 3.2: Table of robustness statistics calculated by Aevol for the best individual with the provided lineage.

We may then compare these statistics for both the control and specific condition we wish to compare (e.g. mutation up). Because this data is somewhat noisy (i.e. the values for these statistics jumps up and down) we will use box and whisker plots to show the overall spread.

Evolvability

In Aevol, evolvability is calculated by generating a large set of offspring for a specific individual (one whose `lineage` was generated) at regular periods along their lineage and then determining the number of "positive offspring". Positive offspring are defined as those whose fitness is greater than their parent's. The evolvability of an individual is then the sum total of all improvement of all of the beneficial offspring, i.e.:

$$\text{evolvability} = \frac{|\text{positive offspring of } i|}{|\text{total offspring of } i|} * \sum \Delta_{fitness}^{\text{positive offspring}}$$

where $\Delta_{fitness}^{\text{positive offspring}}$ is the cumulative sum of the fitness increase for the positive offspring. Thus, evolvability in Aevol accounts for both the likelihood of a positive mutation and the average improvement provided by said

mutation. Practically speaking, to find an organisms evolvability in Aevol one must give the post-treatment `misc_ancestor_robustness` a lineage file and then multiply the fraction of the number of positive offspring (column 2) by the cumulative total of the fitness gap g of the positive offspring (column 10).

3.2.3 Structure

Another strength of Aevol is its ability to analyze changes in the structure of DNA. As with fitness, Aevol produces statistics about individuals and the population at large for many aspects of genome structure, including: the number of coding vs. non-coding bases (i.e. they respectively do or do not code for at least one protein), the average size of coding and non-coding DNAs, the number of genes, the number of "essential" base pairs (i.e. those that are part of a functional coding sequence), etc. Tables 3.3 and Table 3.4 summarize where different statistics are to be found Aevol's output files.

Stat File	
`stat_genes_`⟨`best/global`⟩	`stat_bp_`⟨`best/global`⟩
number of coding RNAs	number of bp not in any CDS
number of non-coding RNAs	number of bp not included in any functional CDS
average size of coding RNAs	number of bp not included in any non-functional CDS
average size of non-coding RNAs	number of bp not included in any RNA
number of functional genes	number of bp not included in any coding RNA
number of non-functional genes	number of bp not included in any non-coding RNA
average size of functional genes	number of non-essential bp
average size of non-functional genes	number of non-essential bp including non-functional genes

Table 3.3: Statistics found in `stat_genes` and `stat_bp` output files from Aevol

"Essential" base pairs are those whose mutation would change the phenotype of the organism.

Stat File	
stat_fitness_⟨best/global⟩	**stat_mutation_⟨best/global⟩**
population size	number of local mutations
fitness	number of chromosomic rearrangements
genome size	number of switches
metabolic error	number of indels
parent's metabolic error	number of duplications
metabolic fitness	number of deletions
secretion error	number of translocations
parent's secretion error	number of inversions
secretion fitness	
amount of compound present in grid-cell	

Table 3.4: Statistics found in `stat_fitness` and `stat_mutation` files from Aevol.

3.3 Expected Results

Given the state of the literature and other experiments, as partially described in Section 2.4, the table below summarizes our expected results.

Chapter 4

Results and Discussion

This chapter presents the results of our experiments as performed according to the description in Chapter 3. The chapter concludes with a discussion of the results and their relation to real-world organisms, as well as potential limitations of our results.

4.1 Results

In this section we present the results of our experiments. We begin by presenting the results of our statistical analysis before moving on to show the results in various plots. It is of minor note that in many of the figures, a rolling window was used to smooth the values, resulting in many of the plots only showing data starting after a few thousand generations (often 10,000). It should be noted, however, that organisms were evolved for 500,000 generations and that the data is complete.

4.1.1 Statistical Analysis

Below in Table we see the results of our p-value calculations for testing whether the results are significant.

4.1.2 Genome Size

In this section we will examine the results of the different conditions, focusing on which, if any, lead to a reduced genome. Figure 4.1 presents the main

findings regarding genome size. In the figure, the blue line represents the control condition and the other colors show the changed conditions: mutation up/down, selection up/down, and population up/down. As can be seen from the figure,the results did not actually vary all that much across the conditions with regard to genome size.

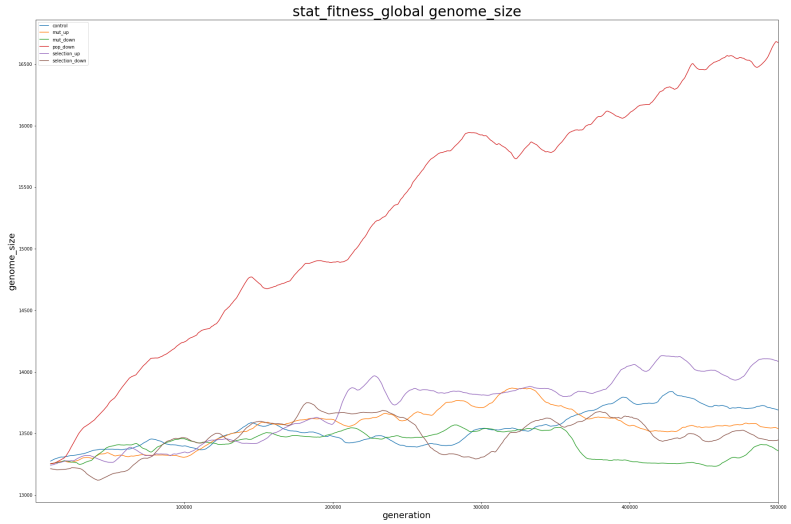

Figure 4.1: Genome size in number of bases of all conditions. Average taken across all five seeds for each condition.

In fact the *population down* condition had a runaway increase in the number of bases, reaching over 16,500 bases, at least a 25% increase over the original wild type's roughly 13,200. Surprisingly, even after 500,000 generations it seems that the upper limit may still not have been reached.

The next obvious observation is that of the remaining conditions, only the *selection up* condition seems to have made much of a significant change with its 11% increase. All of the remaining conditions had a mild increase in genome size.

Also noteworthy is that the mutation down condition appears to have had a steady increase in the number of base pairs until a maximum of just over 14,000 around generation 350,000 before having a fairly sharp decline back to nearly the original size.

4.1.3 Metabolic Error

Figure 4.2 shows the mean metabolic error across all seeds for the whole population over time with respect to each condition.

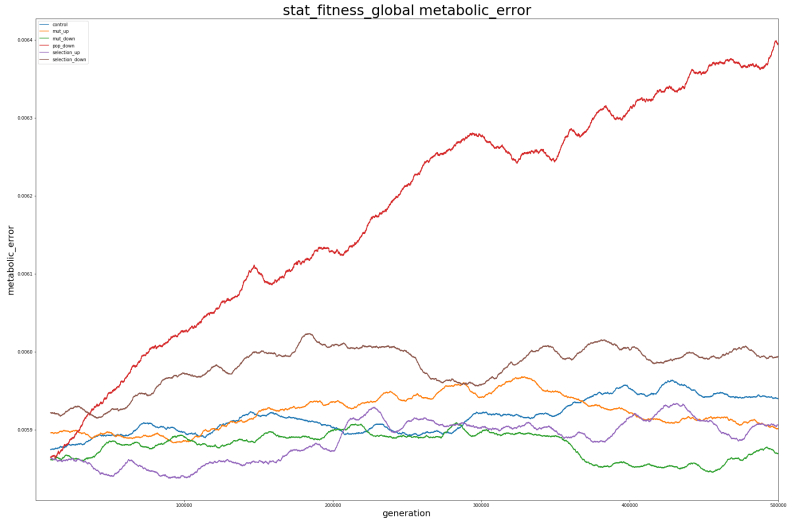

Figure 4.2: Plot of the metabolic error over time for all conditions, average of all seeds

It seems that with the smaller population size, genetic drift may be more strongly at work in continually increasing the gap between the phenotype and environmental function, as the lack of variety inherent in a smaller population causes a cascade of increasingly deleterious consequences.

4.1.4 Genome Structure

In this section we examine the effects of the differing conditions on the structure of the genome as measured by the criteria in Tables 3.3 and 3.4.

Non-coding DNA

One important factor in genome structure is the amount of non-coding DNA, i.e. the number of bases which are part of a genome but which do not encode protein sequences. In the case of aevol, this means

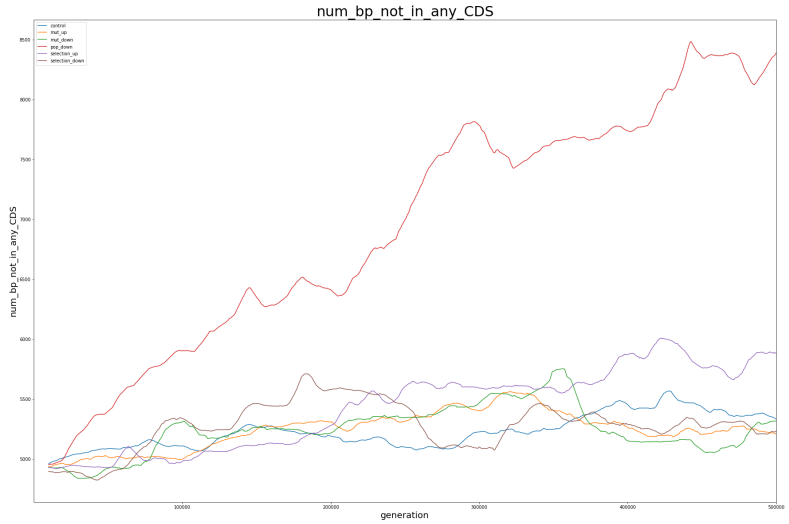

Figure 4.3: Plot of non-coding DNA over time for all conditions, average of all seeds.

Number of Genes

Figure 4.4 below illustrates the mean number of genes across the population for each condition.

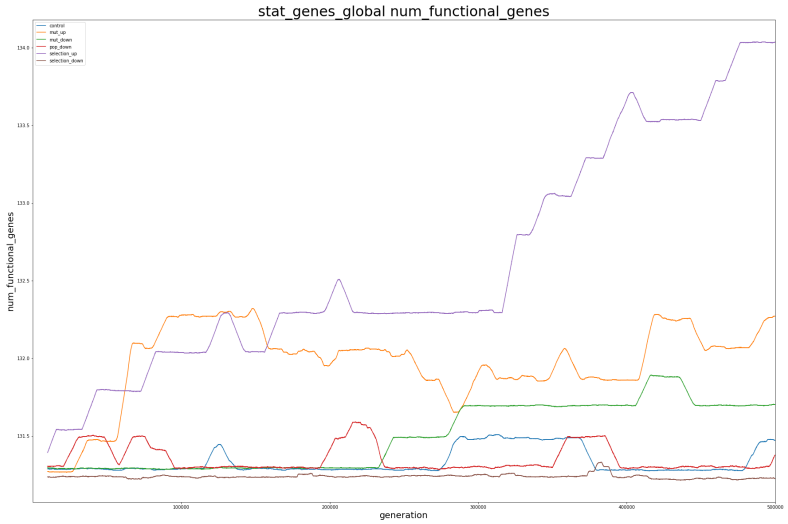

Figure 4.4: Plot showing the mean number of functional genes over time across all seeds.

As seen in the figure, the increased selection condition showed the greatest increase in the number of functional genes in the whole population, about 3% (130 to 134). Interestingly, the selection down condition did not change the number of genes at all, which is to be expected, since with a lower selection pressure, any newer genes may not be selected for reproduction. Whereas most of the other curves are fairly flat, the mutation up condition fluctuates relatively rapidly, owing to the quick increase and decrease in the number of base pairs.

Average Size of Functional Genes

Figure 4.5 shows the average number of base pairs for each functional gene for the best individual, mapped out over the 500,000 generations.

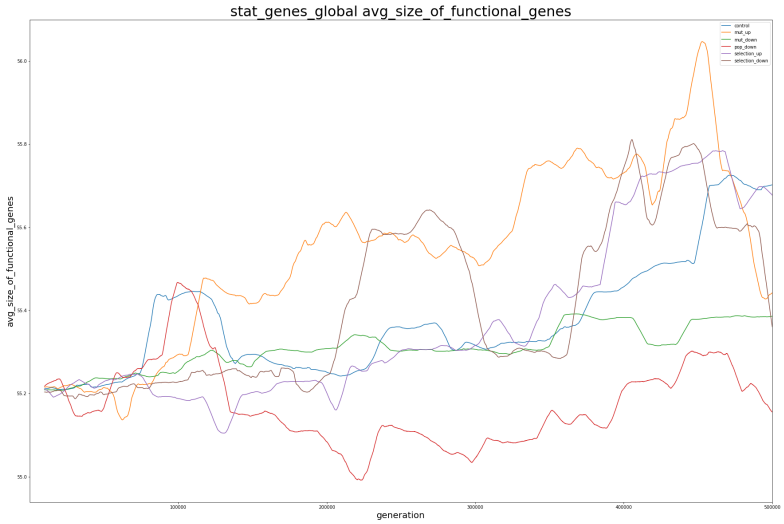

Figure 4.5: Plot showing the average size of functional genes over time for all seeds.

The population down condition continues to be the outlier in terms of genome structure, with the average size of the functional genes remaining noticeably lower than for any other condition. It is worth pointing out, however, that the difference between the smallest average and the largest average is only 1 base pair.

4.1.5 Evolvability

In Figure 4.6 below, we see the results of the experiments on evolvability for the best individual's (at generation 500,000) lineage.

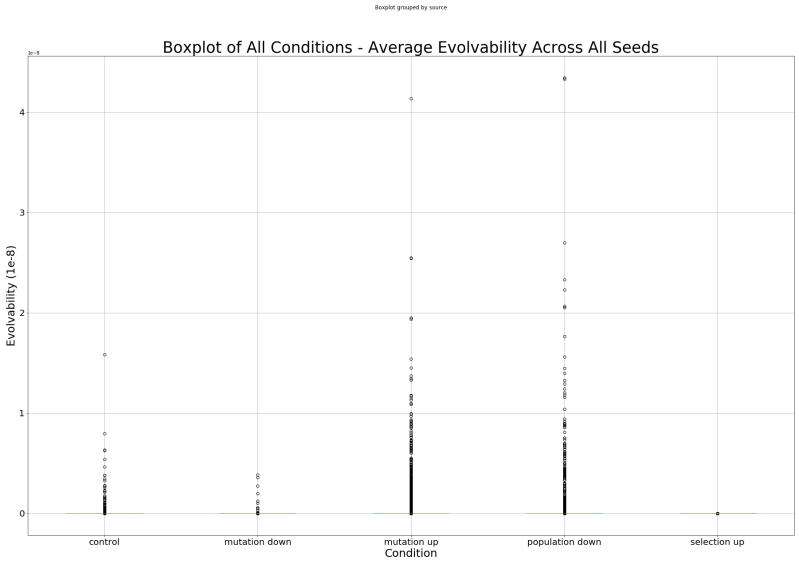

Figure 4.6: A box plot showing the mean evolvability spread of all seeds and all conditions. Higher numbers are more evolvable.

The figure illustrates that, for each condition, overall the best individual still had quite low evolvability. For the selection up condition, however, the deviation from zero was even smaller, as illustrated in Table 4.1 below, which provides the mean and standard deviation of the evolvability of the best individual in each condition.

	mean	**standard deviation**
control	2.035523813975702e-11	3.2787490312081233e-10
μ_{up}	9.97655150597127e-11	8.41749150634588e-10
μ_{down}	6.064697990806935e-12	1.2471674281540128e-10
k_{up}	1.9498939718794653e-15	6.394659146405824e-14
k_{down}		
N_{up}	8.837632116681012e-11	1.0799303682398726e-09
N_{down}		

Table 4.1: Table illustrating the mean and standard deviation of the evolvability for each condition. μ is the mutation rate, k is the selection rate, and N is the population size.

4.1.6 Robustness

Recall that robustness is measured by the fraction of neutral mutants of an individual. In the following figure, we see a bar plot showing the spread of neutral mutants for the best individual for the control condition as well as the six variations.

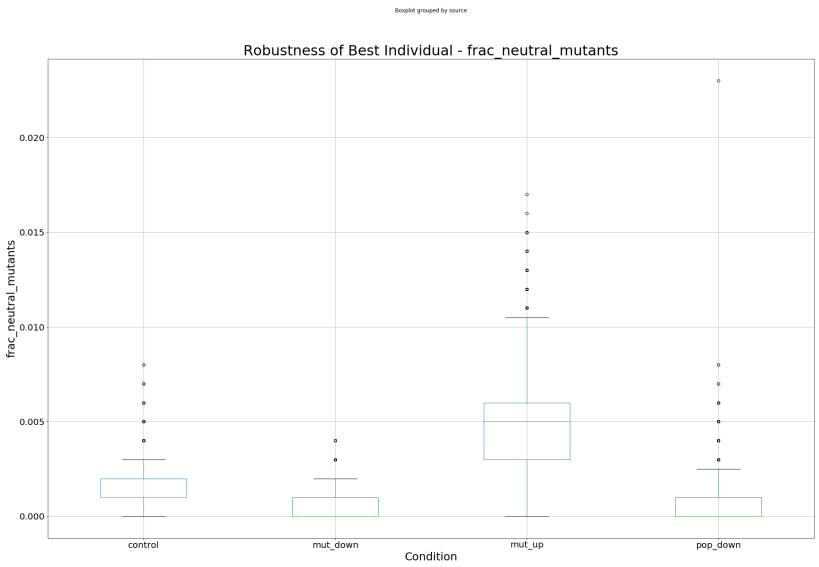

Figure 4.7: Bar graph showing the spread of neutral mutants for the best individual at generation 500,000, all conditions.

4.2 Discussion

4.2.1 Relation to Real-World Biological Entities

4.2.2 Limitations of Results

One limitation to consider is that only one parameter varied per condition. It may be possible that it is only under a combination of conditions (e.g. low selection *and* high mutation rates) does reductive evolution occur.

Another limitation is that the environments did not vary in our experiments. This could potentially have a large effect on robustness and evolvability, which are strong influencers of reductive evolution.

Aevol as a modeling software is limited in that it relies, like all models, on several simplifications. The population sizes tested here, even in the population up condition, are still much smaller than would be found in real world populations.

Chapter 5

Conclusion and Future Work

In this section we will summarize our overall conclusions drawn from the work and highlight a few possibilities for further research.

5.1 Conclusion

5.2 Future work

Bibliography

[1] Bérénice Batut, David P. Parsons, Stephan Fischer, Guillaume Beslon, and Carole Knibbe. In silico experimental evolution: a tool to test evolutionary scenarios. *BMC bioinformatics*, 14 Suppl 15:S11, 2013.

[2] Guillaume Beslon, Vincent F Liard, and Santiago F Elena. Testing evolution predictability using the aevol software. 16th international meeting of the European Society of Evolutionary Biology (ESEB 2017) , August 2017. Poster.

[3] Clarence FG Castillo and Maurice HT Ling. Digital organism simulation environment (dose): A library for ecologically-based in silico experimental evolution. *Advances in Computer Science : an International Journal*, 3(1):44–50, 2014.

[4] Isabel Gordo and Brian Charlesworth. On the speed of muller's ratchet. *Genetics*, 156(4):2137–2140, 2000.

[5] Carole Knibbe. *Evolution of genome structure by indirect selection of the mutational variability: a computational approach.* Theses, INSA de Lyon, December 2006.

[6] Carole Knibbe, Antoine Coulon, Olivier Mazet, Jean-Michel Fayard, and Guillaume Beslon. A long-term evolutionary pressure on the amount of noncoding dna. *Molecular Biology and Evolution*, 24(10):2344–2353, 08 2007.

[7] Sanna Koskiniemi, Song Sun, Otto G Berg, and Dan I Andersson. Selection-driven gene loss in bacteria. *PLoS genetics*, 8(6), 2012.

[8] Vincent Liard, David Parsons, Jonathan Rouzaud-Cornabas, and Guillaume Beslon. The complexity ratchet: Stronger than selection, weaker than robustness. *Artificial Life Conference Proceedings*, 1(30):250–257, 2018.

[9] Octavio Miramontes, Franois Taddei, Ariel B. Lindner, Antoine Frenoy, and Dusan Misevic. Shape matters in cooperation. *Artificial Life Conference Proceedings*, 1(28):340–341, 2016.

[10] Vadim Mozhayskiy and Ilias Tagkopoulos. Microbial evolution in vivo and in silico: methods and applications. *Integrative Biology*, 5(2):262–277, 10 2012.

[11] C. Ofria and C. O. Wilke. Avida: A software platform for research in computational evolutionary biology. *Artificial Life*, 10(2):191–229, 2004.

[12] T. S. Ray and J. Hart. Evolution of differentiated multi-threaded digital organisms. In *Proceedings 1999 IEEE/RSJ International Conference on Intelligent Robots and Systems. Human and Environment Friendly Robots with High Intelligence and Emotional Quotients (Cat. No.99CH36289)*, volume 1, pages 1–10 vol.1, 1999.

[13] Jacob Rutten, Paulien Hogeweg, and Guillaume Beslon. Adapting the engine to the fuel: mutator populations can reduce the mutational load by reorganizing their genome structure. *BMC Evolutionary Biology*, 19, 12 2019.

[14] Yolanda Sanchez-Dehesa, Loïc Cerf, Jose Maria Pena, Jean-François Boulicaut, and Guillaume Beslon. Artificial Regulatory Networks Evolution. In *Proc 1st Int Workshop on Machine Learning for Systems Biology MLSB 07*, pages 47–52, Evry, France, September 2007.

[15] Zhiyi Sun and Jeffrey Blanchard. Strong genome-wide selection early in the evolution of prochlorococcus resulted in a reduced genome through the loss of a large number of small effect genes. *PloS one*, 9:e88837, 03 2014.

[16] Andreas Wagner. Robustness and evolvability: a paradox resolved. *Proceedings of the Royal Society B: Biological Sciences*, 275(1630):91–100, 2008.