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

List of Figures

List of Tables

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*[7]. 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-

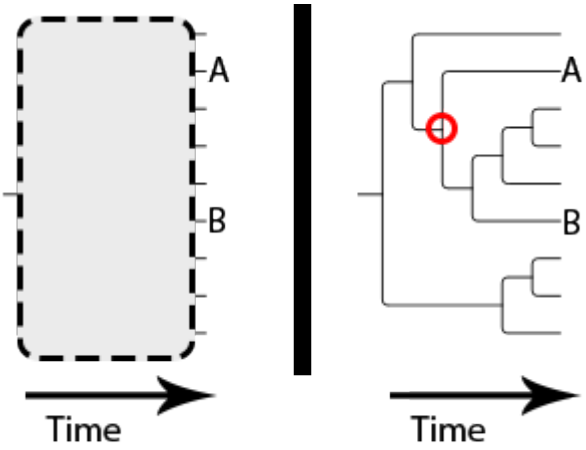

Figure 1.1: An illustration of unknown phylogeny. Since the phylogenetic information under the shaded box is typically not known, the point of divergence (red circle) can't be determined.

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 in silico evolution in general and Aevol in particular. Chapter 3 describes our experimental setup. Chapter 4 provides the results and analysis of the experiments, and Chapter 5 presents our conclusions.

Chapter 2

Background

As discussed in Chapter 1, reductive evolution is a

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 recon-

struction of the system from any given point, allowing one to easily create a record of events, including phylogenetic trees.

Why use Aevol

The in silico tool *Aevol* was developed in the early 2000s 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 [3], examining the role of mutators in reorganizing the genome in order to overcome mutational load [5], examining the effects of population shape on levels of cooperation [4], modeling regulatory networks [6] and more.

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

2.2 Aevol

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.). The population size N is kept constant and a record of each generation is kept so that the phylogenetic lineages can be recreated.

2.2.1 Aevol's Architecture

Aevol's three steps—decoding the genome, selection, and reproduction—will be examined in more detail in this section. These steps are illustrated in Figure 2.2.1.

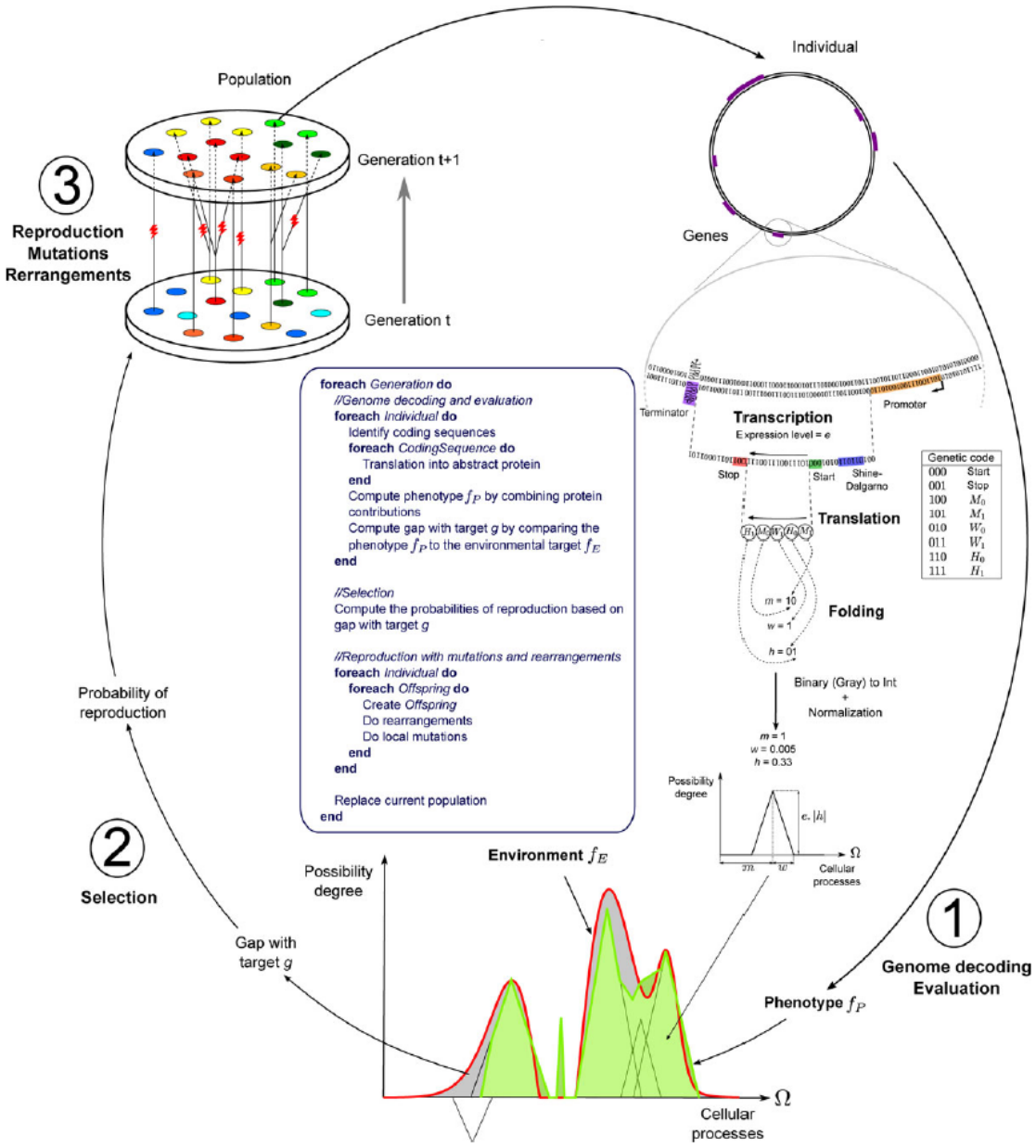

Figure 2.1: Overview of Aevol's architecture, from [1]

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. Terminators 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 2.2.1: 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 $\Omega = [a, b] \subset \mathbb{R}$. 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,

[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`

h) is assigned two codons in the genetic code (see Figure 2.2.1), 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 are then normalized to be in the proper range. w's range is specified in the parameter file (`MAX_TRIANGLE_WIDTH`), h must be in the range $[-1, +1]$ (indicating that both activating and inhibitive processes are allowed) and m must be in the range $[0, 1]$ (the range of Ω).

Given multiple coding sequences in a genome, several triangles 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]. In short, the activating proteins are added up, as are the inhibiting proteins, and the difference between these two totals represents the final function f_P.

Selection

After the genome is decoded, the organisms are tested for their fitness. Fitness in Aevol is defined as the gap between the phenotype of a sequence f_P and the environmental target function f_E, as illustrated in Figure 2.2.1. This environmental target function f_E is a user-determined set of Gaussians which are specified in a parameter file. 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: $g = \int_a^b f_E(x) - f_P(x)dx$.

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 reproduction for each organism is proportionate to its fitness, namely $P(\text{reproduction}) = exp(-k * g)$, where k is a user-definable parameter which determines the selection intensity and g is the metabolic error.

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

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 by drawing from a multinomial distribution. 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 the driving force in evolution occurs, namely the possibility for mutation, indels, and frameshifts. 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 events shuffled randomly.

2.2.2 Aevol's Statistics and Post-Treatments

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 factors. 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.

In the following two subsections we will examine evolvability and robustness, as these two factors play a major role in reductive evolution.

Evolvability

Evolvability is usually defined as the ability of a system (in this case an organism) to evolve. In other words, "if mutations in it can produce heritable

phenotypic variation"[8]. 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.

It is 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.

Robustness

aevol_misc_ancestor_mutagenesis

This post-treatment generates and analyses mutants for the provided lineage. Specifically, this program generates evolvability, robustness, and antirobustness statistics for the mutants.

aevol_misc_ancestor_robustness

Generates mutants for a given lineage and analyzes their robustness, providing several statistics:

aevol_misc_ancestor_stats

Analyzes a lineage and produces the following outputs:

aevol_misc_coalescence

Prints coalescence statistics for a given lineage.

aevol_misc_create_eps

Creates a directory, analysis-generation_XXXXX with several EPS files.The EPS files are as follows:

aevol_misc_extract

Extracts the genotype and/or data about the phenotype of individuals in the provided population and write them into text files for easy parsing by programs such as MATLAB. The program lets you specify whether you want the sequence, the triangle data, or both. By default (i.e. with no parameters) gives just the sequence, into a file called "sequence".

aevol_misc_lineage

Using the tree files, recreates the lineage of an individual. By default, this is done for the best individual, but another individual can be specified by ID number or rank (-i or -r, respectively).

aevol_misc_mutagenisis

This generates and analyzes mutations for an individual from a population backup (as opposed to a lineage file for ancestor_mutagenesis). One can specify point mutations, small indels, duplications, large deletion, translocations, or inversions.

aevol_misc_protein_map

Creates a CSV file which analyzes the proteins generated by the specified lineage. The CSV file specifies:

aevol_misc_robustness

Calculates replication statistics for a given individual (specified by either ID or rank) at a given time. Specifically, the following information is saved in a new directory (analysis-generation_XXXXXXXXXX):

2.3 Changeable Factors

2.3.1 Fitness

2.3.2 Evolvability

2.3.3 Robustness

2.3.4 Structure

Percent Coding vs. Non-Coding DNA

Number of Genes

2.4 Related Work

Chapter 3

Methods

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

3.1 Contributions

3.2 Experimental Designs

3.3 Expected Results

Chapter 4

Experiments, Results, and Discussion

4.1 Experiments

In this section we describe the experiments generally.

4.2 Evaluation Strategy

In this section we describe how we evaluated the experiments, our metrics, etc.

4.2.1 Robustness

4.2.2 Evolvability

4.2.3 Coalescing Time

4.3 Experiment Results

In this section we describe the results of our experiments.

4.3.1 Discussion

4.3.2 Relation to Real-World Biological Entities

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] 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.

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

[5] 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.

[6] 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.

[7] 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.

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