# Simulating sequence evolution for *in silico* experimentation

Sofia Lima, Emma Wu, Sriram Kidambi, Yejie Yun, Mirudhula Mukundan

## 1 Introduction

Conducting physical experiments in order to analyze sequence evolution requires a considerable amount of time and resources. The purpose of our project will be to construct an *in silico* experimentation tool to test scenarios of sequence evolution to better our understanding of evolutionary sequence effects. Creating this *in silico* platform will be important for analyzing individual effect of evolutionary factors (e.g. selection strength) on a population of simulated organisms. Our bevol platform will follow the structure of aevol (*a*rtificial *e*volution; an open-source package written in C++ which is currently unstable) by simulating the variation-reproduction cycle with reasonable simplifications as described by Batut *et al.* in 2013 [1]. With this illustrated approach towards testing sequence evolution *in silico*, we will scrutinize simplifications of the system we are modeling, and compare these with aevol.

In this project, we will establish a baseline model by analyzing genome size in two environments with different selection pressures, similarly to Batut *et al.*[1], such that we should see a decrease in genome size under relaxed selection (see Figure 1 in Appendix). We hope to replicate this figure and expand the *in silico* application by comparing this effect in two different simulated "species" where one has a starting genome size on the order of 100 times that of the other. Other extensions include chromosomal rearrangements during mutation or horizontal gene transfer during reproduction and parallelization on a double stranded chromosome if time permits. We expect our biggest challenge to be designing the pattern-finding algorithm where we locate promoter sequences during the transcription step of our simulation.

## 2 Background

Simulation methods have been developed to model evolutionary scenarios. For instance, Bacmeta is a simulator of genomic evolution in bacterial metapopulations [4]. It incorporates stochastic simulation of neutral evolution within a large collection of interconnected bacteria with an adjustable connectivity network. This model allows the user to specify connectivity between sub-populations of bacteria, incorporating connectivity which is hard to model in phylogeny reconstruction methods.

Aevol simulates sequence evolution of artificial organisms, and allow for *in silico* experimentation of different evolutionary scenarios. The paper discusses how this *in silico* platform can explain the reductive evolution observed in biological circumstances such as endosymbiosis and closely related marine bacteria. However, this package is unstable and we were unable to actually run this program due to segmentation failure. In this project, we try to design our own simulator for virtual populations in python. Our platform differs from that of aevol in that we will not include some of the auxillary programs for things such as visualizing lineage, but rather focus on modeling the reductive evolution scenario as a function of genome size.

# 3 Methods

## 3.1 Overview

Our "bevol" platform will be structured very similarly to aevol (see Figure 1 in Batut *et al*[1]). Our simulation will model transcription, translation, fitness and selection, and mutation. Each artificial organism will consist of a single chromosome with binary nucleotides containing coding and noncoding regions. We transcribe and translate each genome into a set of arbitrary "cellular process" ranging from 0.0 to 1.0. Fitness is measured by comparing the phenotype represented by $f_p$ of each individual to an arbitrary phenotype needed to survive the pre-determined environment $f_e$. Selection will occur by drawing from a multinomal distribution with parameters defined by the fitnesses such that each generation has constant size N. Reproduction of the N selected individuals is asexual. When a chromosome is replicated, it can undergo various types of mutations occurring randomly at pre-defined rates. We can adjust parameters including mutation rates, and $f_e$ to observe and explain genome shrinkage.

## 3.2 Protocol

### 3.2.1 Transcribing and Translating DNA code to protein (Sofia)

Initially, the genome will be randomly generated with at least one coding region. A loose form of transcription will occur between 22 bp promoters and stop codons. We will assign an expression level to the transcript $e = 1 - \frac{d}{1+d_{max}}$ where $d$ represents the hamming distance between the promoter and a pre-defined consensus. The artificial genetic code will used be sequentially translate each 3 bp codon into one of 6 possible amino acids M0, M1, W0, W1, H0, H1, or the start or stop codon. This sequence of amino acids will then be used to compute the phenotypic contribution for each protein which may be inhibitory.

Given the sequence of amino acids, we will calculate $m$, $w$, and $h$, where the protein may be represented as a triangle graph as a function of these values. $m$ represents the mean "cellular process" of the protein, $w$ represents the range of pleiotropy that this protein exhibits, and $h$ represents the efficiency of the protein.

In computational terms, the codons form the Gray codes of the three parameters $m$, $w$, and $h$. For example, if the amino acid sequence is M1,H0,W1,M0,H1, then the Gray code for $m$ is 10, for $w$ is 1, and for $h$ is 01. $w$ is then normalized by multiplying by $w * \frac{w_{max}}{2^{n_w}-1}$ where $n_w$ is the number of W0 or W1 in the sequence; $m$ is normalized similarly between 0 and 1 and $h$ between -1 and 1. With these values of $m$, $w$, and $h$ defining each protein, the global phenotype of the individual may be calculated.

### 3.2.2 Decoding phenotype from proteins (Emma)

Given the set of phenotypic contributions from each protein, the global functional capabilities of the individual $f_p$ will be calculated. $f_p$ represents the phenotype of the individual where several proteins may overlap. Similarly to aevol, this will be represented as

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

such that $f_i$ is the possibility distribution of the $i$-th activator protein with $h > 0$, and $f_j$ is the possibility distribution of the $j$-th inhibitory protein with $h < 0$.

### 3.2.3 Computing Fitness (Sriram)

Given the calculated phenotype, the fitness will be calculated. Similar to aevol, we will compute a gap $g = \int |f_e - f_p|$, where $f_e$ represents a possibility distribution which we will pre-define as the sum of three gaussian functions. In biological terms, $f_e$ represents the optimal degree of possibility for each cellular process to survive in the environment.

### 3.2.4 Selection (Yejie)

The probability of reproduction is $\frac{e^{-kg}}{\sum_{i=1}^{N} e^{-kg_i}}$ where k determines the steepness of the selection coefficient distribution. Selection is then performed such that new generation is drawn from the multinomial distribution representing the number of offspring each individual of the parent generation produces

with parameters $(N, (\frac{e^{-kg_1}}{\sum_{i=1}^{N} e^{-kg_i}}, ..., \frac{e^{-kg_N}}{\sum_{i=1}^{N} e^{-kg_i}}))$.

#### 3.2.5 Reproduction with Mutation (Mirudhula)

On every round of reproduction, various types of mutations can occur, that include point mutation, small and large insertions, small and large deletions, duplication and translocations. Each of these seven types can be considered to have a per-position mutation rate. The number of mutations for each type could be drawn from independent uniform distributions, except for large deletions and translocations which is drawn from a Binomial Law.

Albeit, on each round, these mutations seem to be occuring randomly and seems quite unlike what happens in nature, where mutation rates are intrinsically optimized when bacteria evolve in different environments [2]. A good simulator should then have a system which optimizes this mutation rate. We will be modeling our mutations off a previous study [3], rewarding increased fitness over consecutive generations with an optimized mutation rate. As an extended study, a close comparison between such an optimized model and a more general random model may help in understanding the atypical evolutionary trajectory involved with changing mutation rates.

## 4 Timeline of the project

We will continue to meet weekly with the goal of having all of our individual parts completed by November 16 (3 weeks from now). We will then run a full 2-step simulation together at our meeting on November 18. We will recreate the main figure from Batut *et al.* by our internal meeting on December 2 (5 weeks from now). We will then run our experiments, generate data, and create figures and compile a slide deck by the class presentation on December 6. We will then write a report by the due date December 15.

## References


[1] Batut, B., Parsons, D. P., Fischer, S., Beslon, G., and Knibbe, C. In silico experimental evolution: a tool to test evolutionary scenarios. In *BMC bioinformatics* (2013), vol. 14, Springer, pp. 1–11.

[2] Loh, E., Salk, J. J., and Loeb, L. A. Optimization of dna polymerase mutation rates during bacterial evolution. *Proceedings of the National Academy of Sciences 107*, 3 (2010), 1154–1159.

[3] Paz-y Miño, C., Espinosa, A., Bai, C. Y., et al. The jackprot simulation couples mutation rate with natural selection to illustrate how protein evolution is not random. *Evolution: Education and Outreach 4*, 3 (2011), 502–514.

[4] Sipola, A., Marttinen, P., and Corander, J. Bacmeta: simulator for genomic evolution in bacterial metapopulations. *Bioinformatics 34*, 13 (02 2018), 2308–2310.

# Appendix

Relevant Background Figures and Tables

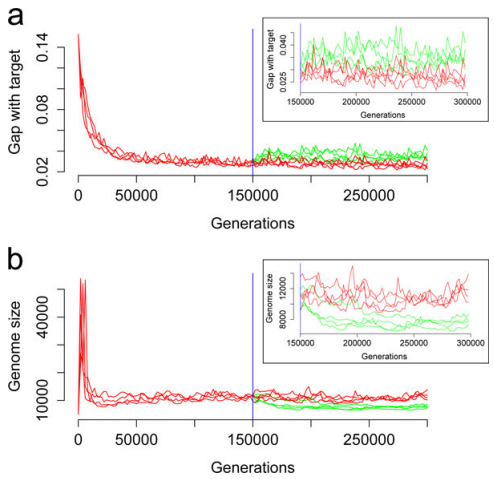


Figure 1: Figure 2 from Batut *et al.* summarizing the results of their in silico experiment analyzing the effect of selective pressure on genome fitness and size (k=250 for the relaxed environment in green; k=750 in red). You can see that under relaxed selective pressure, the population has lower fitness (upper panel) and a reduced genome size (lower panel). The paper discusses how this *in silico* platform can explain the reductive evolution observed in biological circumstances such as endosymbiosis and closely related marine bacteria. Interestingly, in Figure 3 of this paper, they show how most of the genome loss occurs in non-coding regions.

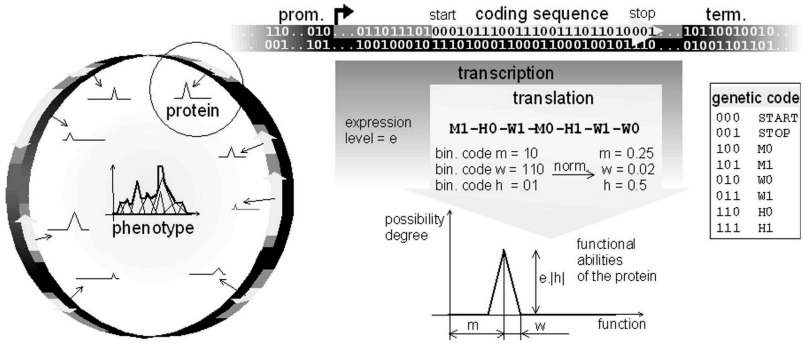


**Figure 1:** Overview of the *aevol* model.

Figure 2: Translation of DNA and decoding phenotype.

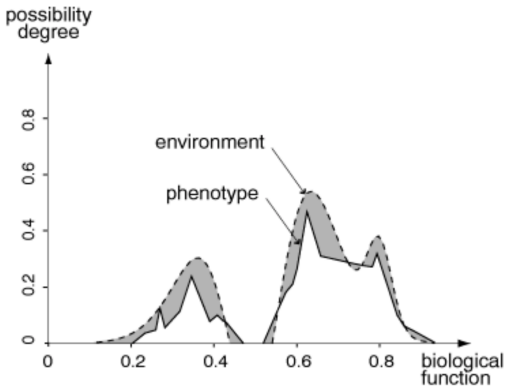


**Figure 2 :** Measuring the adaptation of an individual. Dashed curve: environmental distribution $f_E$. Solid curve: phenotypic distribution $f_P$ (resulting profile after combining all proteins). Filled area: gap $g$.

Figure 3: Computation of fitness from the phenotypic distribution, which is important for the selection and mutation step.

| Parameter | Symbol | Value |
|---|---|---|
| Population size | $N$ | 1,000 |
| Size of the initial (random) genome | $L_{\text{init}}$ | 5,000 base pairs |
| Promoter sequence | - | 01010110011110010010110<br>with up to $d_{\text{max}} = 4$ mismatches |
| Terminator sequences | - | $abcd$ * * * $\overline{d}\,\overline{c}\,\overline{b}\,\overline{a}$ |
| Initiation signal for the translation | - | 011011***000 |
| Termination signal for the translation | - | 001 |
| Genetic code | - | See Figure 1 |
| Global set of "biological functions" | $\Omega$ | [0, 1] |
| Maximal pleiotropy of the proteins | $w_{\text{max}}$ | 0.033 |
| Environmental possibility distribution | $f_{\text{E}}$ | See Figure 2 |
| Selection scheme | - | Linear ranking |
| Selection intensity | $\eta^{+}$ | 1.998 |
| Point mutation rate | $u_{\text{point}}$ | $10^{-5}$ per position |
| Small insertion rate | $u_{\text{smallins}}$ | $10^{-5}$ per position |
| Small deletion rate | $u_{\text{smalldel}}$ | $10^{-5}$ per position |
| Large deletion rate | $u_{\text{largedel}}$ | $10^{-5}$ per position |
| Duplication rate | $u_{\text{duplic}}$ | $10^{-5}$ per position |
| Inversion rate | $u_{\text{inv}}$ | $10^{-5}$ per position |
| Translocation rate | $u_{\text{transloc}}$ | $10^{-5}$ per position |
| Length of small indels | - | Uniform law between 1 and 6 positions |
| Length of rearrangements | - | Uniform law between 1 and $L$ positions |

**Table 1 :** Parameter values used for the run detailed in this section.

Figure 4: Parameters as used in Aevol. A subset of these parameters including $N$, $L_{init}, w_{max}$, $f_e$, and mutation rates will be used in our project.