Simulating sequence evolution for *in silico* experimentation

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

Abstract

Here, we present Bevol, an *in silico* sequence evolution simulator that can test the effect of selective pressure on genome size. The simulator is based on artificial populations with single strand chromosomes using binary genetic code. Selective pressure is defined as a single parameter k that users can alter to simulate different evolutionary scenarios. Arbitrary cellular processes and proteins are defined to calculate fitness from parameter k. The resulting simulations showed that populations under high selective pressure had a larger genome size compared to those with low selective pressure. Although results revealed that populations with low selective pressure exhibited a decrease in genome size over the course of evolution, in some cases populations with high selective pressure had minimal change in genome size.

1 Introduction

Understanding genome evolution is essential to expanding our knowledge of speciation, gene expression, and many more insights to how different species have evolved over time. In particular, the field of bacterial evolution is a unique field in evolutionary biology. More specifically, bacterial species are often the ad hoc species for studying genomic evolution for their abilities to evolve and reproduce quickly. However, the characteristics that make bacterial evolution unique also poses in addition to recreating their environmental conditions make it a challenge in modeling and understanding their evolution. Furthermore, modeling bacterial evolution *in vitro* can be extremely resource and time consuming, hence we turn to *in silico* sequence evolution experiments.

Examining the comparative genomics of oceanic bacterial species and endosymbionts is an interesting challenge in particular as the genomes of these species have undergone reductive evolution despite living in widely variable environments. Our *in silico* platform can allow us to simulate these environments while examining the effects on the bacterial genomes. Current *in silico* platforms like Aevol have shown initial results for comparing reductive evolution in two simulated "species" in variable enviroments [1]. We expand on their *in silico* application by comparing this effect in two different simulated "species" where one has a starting genome size on the twice the size of the other and further additional environments.

Overall, our goal is create a platform, Bevol, that simulates sequence evolution of artificial organisms, and allow for *in silico* experimentation of different evolutionary scenarios. Creating this *in silico* platform will be important for analyzing individual effect of evolutionary factors (e.g. selection strength) on a population of simulated bacterial organisms. Our bevol platform will follow the structure of aevol by simulating the variation-reproduction cycle with reasonable simplifications as described by Batut *et al.* in 2013 [1]. For the remainder of this paper, we will go into detail about our methodology for the creation of this platform. With aforementioned approach towards testing sequence evolution *in silico*, we will also scrutinize simplifications of the system we are modeling, and compare these with aevol.

2 Methods

2.1 Overview

Closely following the methods used by Batut *et al*[1]), our simulator will follow the steps of transcription, translation, phenotypic computation, selection and mutation that constitute one simulator

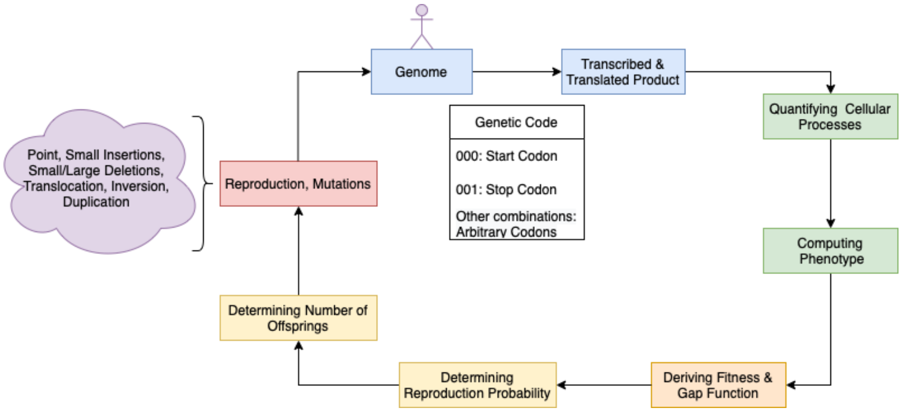

Genome
Transcribed & Translated Product
Quantifying Cellular Processes
Computing Phenotype
Deriving Fitness & Gap Function
Determining Reproduction Probability
Determining Number of Offsprings
Reproduction, Mutations
Point, Small Insertions, Small/Large Deletions, Translocation, Inversion, Duplication
Genetic Code
000: Start Codon
001: Stop Codon
Other combinations: Arbitrary Codons

Figure 1: Overview of the protocol used for developing Bevol

generation of sequence evolution. 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. Every generation, the set of individuals are replaced by a completely new set of offsprings such that the same population size is constant throughout. This helps in observing the effect of selection over multiple bacterial generations in the span of just a few simulator generations due to preferential selection of only the best adapted individuals. The complete methodology is depicted as a flowchart in Fig.1.

2.2 Protocol

2.2.1 Transcribing and Translating DNA code to protein

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 can identify transcripts by the presence of promoters and terminators, following which an expression level is assigned to each transcript, such that $e = 1 - \frac{d}{1+d_{max}}$ where d represents the hamming distance between the promoter and a pre-defined consensus. The artificial genetic code will be used to sequentially translate each of 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 based on the cellular process that each protein is involved in.

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 is calculated.

2.2.2 Decoding phenotype from proteins

Each protein could be responsible for several cellular processes, which can be defined by a certain degree of possibility for each process, represented as $f_i(x)$, where i represents a particular protein, and x is the cellular process. By using a piecewise-linear distribution over the triangles and its characteristic m, w, and h, obtained from the previous step, we can calculate a series of these $f_i(x)$ values. Given the set of phenotypic contributions from each protein, the global functional capabilities of a particular cellular process over multiple proteins is calculated as $f_p(x) = max(min(\sum_i f_i(x), 1) - min(\sum_j f_j(x), 1), 0)$, where 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 $h < 0$.

2.2.3 Computing Fitness

Fitness is calculated relative to an ideal environment with an possibility distribution, f_E, for each cellular process. This is considered to be an optimal set of values where the organisms are free to reproduce under no selective pressure, and is preset at the beginning of the simulation, by sampling from a sum of three gaussian distributions. The fitness is then obtained by calculating the difference between optimal and actual degree of possibility for each cellular process and summing over them all as: $g = \int_0^1 |f_e - f_p|$, where g represents the gap between actual and optimal values.

2.2.4 Selection

The gap function represents the adaptive capability of an organism. So, this can help with calculating the reproductive strength of an organism as $\frac{e^{-kg}}{\sum_{i=1}^{N} e^{-kg_i}}$ where k is the external factor deciding the selective pressure of the environment and drives which individual is capable of reproducing the most under that coefficient of selection. From this, the number of offsprings from a particular organism can be determined by sampling from a multinomial distribution as,

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

2.2.5 Reproduction with Mutation

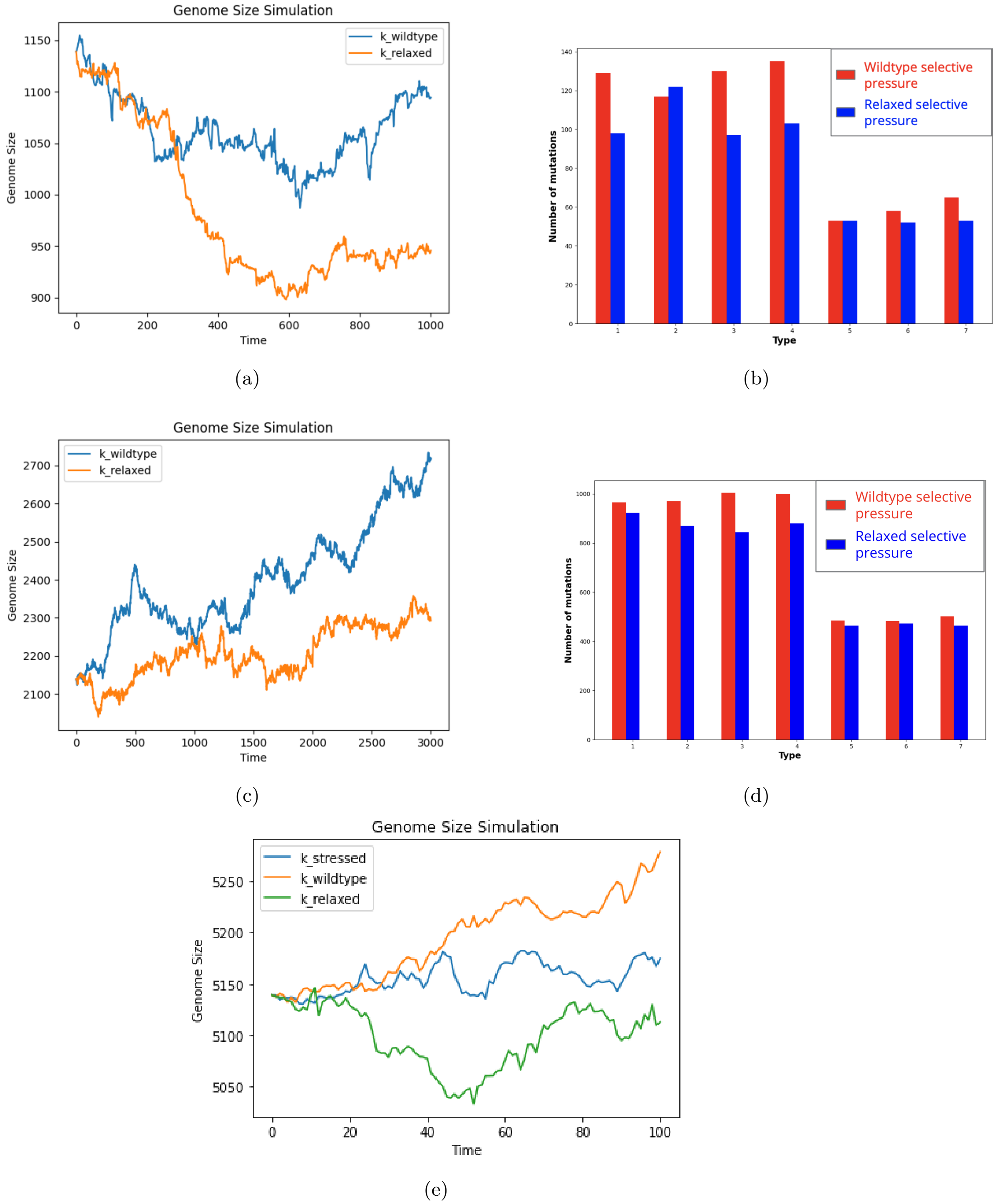
On every round of reproduction, various types of mutations can occur, that include point mutation, small and large insertions, small and large deletions, duplication and translocations. Each of these seven types can be considered to have a per-position mutation rate. The number of mutations for each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

Going into detail, for a point distribution the binary code for a randomly selected position is inverted, whereas for small insertions and deletions, a random sequence of 1-6 bp is inserted/deleted at a random position. Large deletions is different from small deletions where a larger sequence is deleted (around 15-20 bp). Lastly, for duplication and translocation, s section of 15-20 bp sequence that is randomly chosen is moved or copied to a separate location that is drawn from a uniform distribution.

3 Results

To model the effects of selective pressure on genome evolution, three simulations were conducted, illustrating wildtype, relaxed, and stressed selective pressure. Our baseline model shows that, over the course of hundreds of generations, the population genome size decreases under relaxed selective pressure (Figure 2a). In accordance to our expectations, the decrease in selective pressure renders some bacterial genes as non-essential, and therefore as the population size decreases, so does the genome size.

Another round of simulation was conducted to observe the effects that selective pressure could have on different species, i.e. increasing the genome size and the number of generations (Fig.2c). The increase in population size and generations had a similar effect as seen in Figure 1. The genome size under relaxed selective pressure is much smaller compared to the size under the wildtype conditions. The decrease may be attributed to the idea that individuals with a smaller genome size have greater fitness under relaxed pressure and therefore had more offspring. But, we also do not expect to see similar kind of simulations on every run because as per our understanding of sequence evolution, selective pressure may not be the only criteria that is driving genome reduction, and not having a reduction on every



(a)

(b)

(c)

(d)

(e)

Figure 2: Bevol simulation tests: (a) Baseline Simulation. Parameter settings: Number of generations = 1000, Genome Size = 1000 bp, Number of individuals = 5, k_wildtype = 750, k_relaxed = 250, (b) Cumulative number of mutations across wildtype and relaxed simulation for baseline simulation (See end for more details), (c) Simulation with Increased genome size and number of generations. Parameter settings: Number of generations = 3000, Genome Size = 2000 bp, Number of individuals = 20, k_wildtype = 750, k_relaxed = 250, (d) Cumulative number of mutations across wildtype and relaxed simulation for second simulation (See end for more details), (e) Simulation of Stressed condition. Parameter settings: Number of generations = 100, Genome Size = 5000 bp, Number of individuals = 10, k_wildtype = 9, k_relaxed = 3, k_stressed = 15. Details on mutation numbers plot: Red: Wildtype; Blue: Relaxed; Key along x-axis as mutation types: 1. Large Deletion, 2. Inversion, 3. Duplication, 4. Translocation, 5. Point Mutation, 6. Small Insertion, 7. Small Deletion.

run is clearly an indication of the same. Furthermore, we wanted to observe this genomic reduction had on the cumulative mutation numbers at the end of each run. Therefore, the number of mutation events of each type from the winning parental candidate that was the most fit or, alternatively had the most offsprings, in each generation was recorded (See Fig.2b, 2d). From this, we observed that relaxed selective pressure overall had a fewer mutational events than the wildtype. We do not see any direct correlation between these mutational rates and the selective pressure since there seems to be a uniform decrease in the mutational events across all types for relaxed pressure. But, one possible explanation on why there seems to be a reduced number of events in the relaxed case is by drawing a precedent with genetic drift, where organisms undergo genome reduction. It is possible for a high fitness variant to get selected for multiple generations, which may cause other variants to disappear. Since the same variant appears as fit in several generations, we observe a lesser number of mutational events with respect to the winning candidate in the relaxed pressure case.

A final experimental run was conducted to observe the effect that stressed selective pressure had on the genome size (Fig.2e). Consistent with our previous findings, the relaxed and wildtype cases behaved as expected. However, it was interesting to observe that stressed condition did not show a significant difference in genome size across generations. More experimental runs with varying genome size and number of generations could possibly help us understand this phenomenon further.

4 Discussion

This project successfully modeled sequence evolution *in silico*. We were successful in solving a wide variety of computational problems including discrete optimization, namely pattern matching for finding the promoter in transcription, and sampling from a probabilistic distribution for selection as well as for mutagenesis. Our method is not optimized for computational efficiency and could greatly benefit from a more sophisticated substring search algorithm. Further optimization of this platform could also include computing individual fitnesses in parallel, as a form of distributed optimization.

While our methodology differed slightly from that of Aevol, namely in the normalization techniques and algorithm implementation, we can compare our results. We present observations where the genome size of our synthetic organisms reduces in an environment under relaxed selective pressure. However, more simulations and statistical analysis should be conducted in order to make confident conclusions about the relationship between genome size and selective pressure.

This proof-of-concept project shows how such a platform can be used for studying sequence evolution. Our model makes important assumptions in order to compromise for feasibility. The authors of Aevol also discuss the fact that “obviously, working with simulated - false - organisms is the major drawback of this approach.” Using such a platform with real data would be a more reliable method for making important conclusions about biological sequence evolution.

5 Future Work

It may be worthwhile to test the model for other factors responsible for reductive evolution, in other cases like genetic drift, bottlenecks, etc. For this end, mutation rates, addition of pleiotropy elements along others could be varied. Not restricting ourselves to reductive evolution, we could also model metabolic changes in bacteria and adaptations of the organism when exposed to antibiotics. In fact, we could also go beyond bacteria to attempt at simulating viruses and other organisms.

Another interesting future work could be with respect to mutations. For now, on each round, 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 could also attempt to model 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.

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.