Master Thesis

Experiments in In Silico Evolution with Aevol

Brian Davis

Supervisor: Berenice Batut

March 2020

Albert-Ludwigs Universität Freiburg

Bioinformatics Group

Professor Dr. Rolf Backofen

Abstract

In silico evolution is an area of active research in the field of bioinformatics. This method seeks to simulate organisms and their evolution in software, allowing for greater control of the environment, mutation rates, and other variables. In this thesis we use the in silico tool aevol to study reductive evolution.

Contents

List of Figures

List of Tables

Chapter 1

Introduction and Background

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, regulatory abilities, etc. The reasons for this are an area of active research , and one such field of research is *in silico evolution*. 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, potentially leading to 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 lead to a reduced genome in simulated bacteria using the aevol platform.

1.1 Problem Statement

1.1.1 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 gives our conclusions.

In this section we provide some general background information about in silico evolution and aevol, the artificial evolution platform used in our experiments.

Chapter 2

Background

2.1 In Silico Evolution

In vivo experiments, although sometimes more realistic, have their own set of difficulties, including challenging environmental conditions (e.g. simulating the open ocean in a lab) and multiple selection pressures acting on genomes in the real world [1], sometimes making it difficult or impossible to tell which factor(s) lead to which outcome.

In silico evolution simulates organisms in software, thus allowing for far greater control and analysis of the environment and other experimental conditions. Varying strategies can be used to create the organisms (e.g. random genes, explicitly providing a previously-evolved genome, etc.) and then the organisms may interact, reproduce, and evolve, with the fittest individuals surviving to produce the next generation.

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

TODO - Comparison with AEGIS[2], etc.

In the following sections, each of the three steps of aevol will be examined in greater detail.

2.2 aevol

The in silico tool *aevol* was developed to "study the evolution of the size and organization of bacterial genomes in various scenarios"[1]. 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

Decoding the Genome

As mentioned above, in aevol a genome consists of a string of binary characters where 0 is complementary to 1. Each organism in the clonal population has a double-stranded circular genome which is either generated randomly or which was specified as input. 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 010101100111001001010. Terminators are sequences which can form a stem-loop structure with a stem size of 4 bases and a loop length of 3 bases.

Selection

After the genome is decoded, the organisms are tested for their fitness. Fitness is roughly defined as their ability to fulfill a target function.

Reproduction

2.2.2 aevol's Post-Treatments

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.

1. Generation
2. Fraction of positive mutants
3. Fraction of neutral mutants (reproductive robustness)
4. Faction of neutral mutants (mutational robustness)
5. Fraction of negative mutants
6. Cumulative delta-gaps of positive mutants
7. Cumulative delta-gaps of negative mutants
8. Delta-gap for the best mutants
9. Delta-gap for the worst mutant
10. Cumulative delta-fitness of positive mutants
11. Cumulative delta-fitness of negative mutants
12. Delta-fitness for the best mutants
13. Delta-fitness for the worst mutants

aevol_misc_ancestor_robustness

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

1. Generation
2. Fraction of positive offspring

3. Fraction of neutral offspring (aka reproductive robustness)
4. Fraction of neutral mutants (aka mutational robustness)
5. Fraction of negative offspring
6. Cumulation of delta-gaps of positive offspring
7. Cumulation of delta-gaps of negative offspring
8. Delta-gap for the best offspring
9. Delta-gap for the worst offspring
10. Cumulation of delta-fitness of positive offspring
11. Cumum of delta-fitness of negative offspring
12. Delta-fitness for the best offspring
13. Delta-fitness for the worst offspring

aevol_misc_ancestor_stats

Analyzes a lineage and produces the following outputs:

1. non-coding statistics (e.g. ancestor_stats_bp-b000000000-e000010000-i196-r1.out)
2. mutation statistics (e.g. ancestor_stats_mutation-b000000000-e000010000-i196-r1.out)
3. Each generation's M W H (currently not working? – e.g. ancestor_stats_envir-b000000000-e000010000-i196-r1.out)
4. Each generation's genome size and termination number (e.g. ancestor_stats_nb_term-b000000000-e000010000-i196-r1.out)
5. The lineage's fitness statistics
   (a) Generation
   (b) Population size
   (c) Fitness

(d) Genome size
(e) Metabolic error
(f) Parent's metabolic error
(g) Metabolic fitness
(h) Secretion error
(i) Parent's secretion error
(j) Secretion fitness
(k) Amount of compound present in the grid cell
(l) Int probe
(m) Double probe

6. Statistics about the how many genes are in 20 RNA (e.g. ancestor_stats_operons-b000000000-e000010000-i196-r1.out)

7. Individual gene statistics

(a) Generation
(b) Number of coding RNAs
(c) Number of non-coding RNAs
(d) Average size of coding RNAs
(e) Average size of non-coding RNAs
(f) Number of functional genes
(g) Number of non-functional genes
(h) Average size of functional genes
(i) Average size of non-functional genes

8. Rearrangement statistics (e.g. ancestor_stats_rear-b000000000-e000010000-i196-r1.out)

(a) Generation
(b) Actual duplication rate
(c) Actual deletion rate
(d) Actual translocation rate
(e) Actual inversion rate
(f) Average alignment score

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:

- best_genome_with_CDS.eps
- best_genome_with_mRNAs.eps
- best_phenotype.eps
- best_pos_neg_profiles.eps
- best_triangles.eps

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:

1. generation – Which generation is being described
2. protein_id – An identifier for the protein.
3. shine_dal – The location of the Shine-Dalgarno initiator for the protein on the sequence.
4. length – The number of base pairs which encodes the protein.
5. concentration – The "amount" of a protein encoded, specified by the distance between the promoter and the consensus sequence.
6. hamming_dist – The Hamming distance between the promoter sequence and the consensus sequence.
7. dist_next_protein – The distance to the next protein?
8. nb_proteins – The number of proteins?
9. dna_length – The length of the DNA sequence.

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):

1. Parent id – The identifier of this organism's parent.
2. Parent metabolic error – The metabolic error rate of the parent.
3. Parent secretion – The secretion rate of the parent.
4. Mutant metabolic error – The metabolic error rate of the mutant.
5. Mutant secretion – The secretion rate of the mutant.
6. Mutant genome size – The genome size of the mutant.
7. Mutant number of functional genes – The number of functional genes in the mutant.

2.3 Related Work

Chapter 3

Approach

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 Experiment 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] William J. Bradshaw, Arian Šajina, and Dario Riccardo Valenzano. Aegis: An in silico tool to model genome evolution in age-structured populations. *bioRxiv*, 2019.