Aevol 4b: Bridging the gap between artificial life and bioinformatics

Hugo Daudey, David P Parsons, David P. Parsons, Eric Tannier, Vincent Daubin, Bastien Boussau, Vincent Liard, Romain Gallé, Jonathan Rouzaud-Cornabas, Guillaume Beslon

Aevol_4b: Bridging the gap between artificial life and bioinformatics

Hugo Daudey[1], David P. Parsons[1,2,3], Eric Tannier[1,3], Vincent Daubin[1], Bastien Boussau[1], Vincent Liard[4], Romain Gallé[2,3], Jonathan Rouzaud-Cornabas[2,3] and Guillaume Beslon[2,3]

[1]LBBE, CNRS UMR 5558 Laboratoire de Biométrie et Biologie Évolutive, Université de Lyon 1, Villeurbanne, France.
[2]INSA-Lyon, UCBL, Univ. Lumière, École Centrale de Lyon, LIRIS UMR CNRS 5205, Villeurbanne, France
[3] Centre Inria de Lyon, Villeurbanne, France
[4] Université Paris-Saclay, LISN UMR CNRS 9015, Gif-sur-Yvette, France
guillaume.beslon@insa-lyon.fr

Abstract

A common subject: Evolution through a computational lens. Two different communities: on the one hand, artificial life researchers use computational systems to understand emergent evolutionary processes and patterns such as complexity, robustness, evolvability and open-endedness; on the other hand, evolutionary bioinformatics researchers decipher patterns and processes in diverse domains of life on Earth using computational methods based on biological data. Both communities use simulations of living organisms but with different aims, objects, and methods, resulting in disjoint research corpuses. We propose Aevol_4b, an artificial life evolution simulator, and show that the data it produces can be successfully and interestingly processed using bioinformatics methods. This bridges the gap between the two fields and paves the way for fruitful exchanges between artificial life models and bioinformatic analysis methods.

Introduction

Over the last 50 years, computer science has progressively become a natural companion for life sciences. However, the interactions between life sciences and computer science take various forms. On the one hand, *Bioinformatics* focuses on "Research, development, or application of computational tools and approaches for expanding the use of biological, medical, behavioral or health data" (Huerta et al., 2000). On the other hand, *Artificial Life* stresses the use of modeling and simulation in order to study "life as it could be" (Langton, 1988). Clearly, both approaches focus on the same objects (living beings). However, they hardly interact, each having its own conferences and journals and cross-fertilization being the exception rather than the norm. This is all the more surprising given that bioinformatics, like artificial life, often uses similar modeling and simulation approaches as tools for knowledge production, whether to formalize knowledge or to produce test sets for sequence analysis or phylogeny algorithms (Trost et al., 2024). This lack of communication can be exemplified by looking at two articles published in 2012 in the prestigious *Nature Reviews* journal series. (Hoban et al., 2012) was published in *Nature Reviews Genetics* and presents simulation as a tool for population and evolutionary genetics. It is grounded in bioinformatics. (Hindré et al., 2012) was published in *Nature Reviews Microbiology* and presents *in silico* experimental evolution. It is grounded in experimental evolution and artificial life. The former article has 101 references and the later, 158 references. Although both focus on simulation in evolution, not a single one of these 259 references is common to both articles. In other words, there are two independent research corpuses, both pursuing similar objectives (understanding life and evolution), using the same methods and tools, but almost completely ignoring one another.

Our goal in this article is to at least partially bridge the gap between artificial life and bioinformatics by developing simulations grounded in artificial life but able to produce data that can be analyzed by (and be used as benchmarks for) off-the-shelf bioinformatic tools.

We will first try to understand the reasons of the schism that separates bioinformatics from artificial life. This will lead us to propose a new simulator, Aevol_4b, based on the Aevol platform (Knibbe et al., 2007; Liard et al., 2020; Banse et al., 2023) but with the major difference that Aevol_4b genetic sequences comply with not only the structure (as in Aevol) but also the information encoding of real biological genomes, making them fit for analysis by generic bioinformatic tools without adapting these tools to the specifics of the simulator. We will then use Aevol_4b to simulate evolution along a large species tree and show how it is possible to recover the original tree using only the information provided by the final sequences. Finally, we'll look at the prospects opened up by this unprecedented combination of artificial life and bioinformatics.

Simulators vs. Simulators

The Genetic Simulation Resources website[1] (GSR) references over 220 software dedicated to the simulation of genomes and genome evolution (Peng et al., 2013). Although some well known simulation platforms are not listed in the GSR (*e.g. Evolver* (Edgar et al., 2009)), it is quite representative of the models used in bioinformatics and con-

[1]https://surveillance.cancer.gov/genetic-simulation-resources/

tains some of the "best-sellers" in the domain (*e.g. Alf* (Dalquen et al., 2012), *SLiM* (Haller and Messer, 2019) or *SimPhy* (Mallo et al., 2016)). The artificial life counterpart of the GSR, though at a smaller scale, lives on the Alife website [2], where nine simulation platforms for artificial life are listed, including *Avida* (Adami, 2006), the most famous platform in artificial life. Surprisingly, however, none of these platforms are mentioned in the GSR. This clearly shows that the two types of simulators are rooted in different communities and are likely to be based on different design principles.

We would like to propose that one explanation (possibly among others) of the difference between simulators based on bioinformatics and those based on artificial life is that bioinformatic simulators aim primarily at reproducing the *patterns* observed on real genomes, whereas artificial life simulators aim primarily at simulating the *processes* that are at the heart of the evolutionary phenomenon. Doing so, they allow for studying the different patterns that emerge from the combined action of these processes (even though these patterns could be different from what is observed on real genomes). This means that instead of tuning models to resemble empirical genomic data, artificial life simulators mechanistically simulate the processes that produce these empirical data. One might object that some bioinformatic simulators – typically forward-in-time population genetics simulators such as *SLiM* (Haller and Messer, 2019) – include the two main evolutionary processes: variation and population-based selection. However, these two processes are not sufficient to account for evolution. Indeed, all living being also embody a third process, the genotype-to-phenotype mapping. This mapping is classically included in artificial life simulators in order to compute a fitness for the organisms. Bioinformatic simulators however, don't include this mapping and directly assign a selection coefficient s to the mutations, s being drawn from a predefined distribution provided as a model parameter ($s > 0$ corresponding to favorable, $s = 0$ to neutral and $s < 0$ to deleterious mutations).

Although the specifics of the way a genotype is mapped to a phenotype may appear to be a simple modeling choice, it actually has a crucial impact on the evolutionary dynamics. Indeed, the genotype-to-phenotype map indirectly controls the shape of the fitness landscape by determining directly or indirectly the effects of mutations. Roughly, one could distinguish three levels of increasing complexity:

Direct Fitness Effect Mutations are directly assigned a fitness effect drawn from a Distribution of Fitness Effect (DFE). In that case, the fitness landscape metaphor is no longer valid on a global scale since a constant DFE implies a constant local fitness landscape. In particular, there are no fitness peaks or valleys.

Mathematical Mapping The genotype-to-phenotype map is indirectly implemented by a mathematical function defining the fitness effect of the mutations (*e.g.* Fisher's Geometric Model (Fisher, 1930) or NK-Fitness Landscape (Kauffman and Levin, 1987)). In that case the shape of the fitness landscape varies locally (including peaks, valleys and possibly local optima). However, the size and ruggedness of the fitness landscape are constant.

Algorithmic Mapping The genotype-to-phenotype map is implemented by an algorithm interpreting the genomic information (*e.g.* Avida (Adami, 2006) or Aevol (Knibbe et al., 2007)). In that case the fitness landscape has a complex structure with peaks and valleys and a ruggedness that may also vary across macroscopic regions. In some models, the size of the fitness landscape can also vary during evolution (Liard et al., 2020).

From the point of view of artificial life, it could be tempting to criticize bioinformatic simulators for neglecting something as important as the genotype-to-phenotype map. Indeed, bioinformatic approaches, because they use a predefined DFE, can hardly address questions such as the evolution of robustness or evolvability, whereas these questions are typically addressed by artificial life (Wilke et al., 2001; Crombach and Hogeweg, 2008; Liard et al., 2020). However, it's important to keep in mind that simulating genotype-to-phenotype mapping has a significant cost, which is by itself sufficient to justify the approach taken by bioinformatic simulators. First of all, the mapping complexifies the models, making it more difficult to analyze the results. Second, having to compute the phenotypes of all the individuals in a population obviously has a computational cost that limits the complexity of the individuals and the size of the population that can be simulated. On top of that, at least to date and probably for a long time to come, nobody knows how to decode a realistic genotype into a realistic phenotype. In other words, by incorporating a genotype-to-phenotype mapping, artificial life platforms necessarily distance themselves from real genomes, let alone real phenotypes!

There lies the gap: bioinformatic simulators aim at studying the evolution of genomic patterns but, to do so, they are obliged to dispense with genotype-to-phenotype mapping. Artificial life simulators include a genotype-to-phenotype decoding process by essence, but, in turn, they cannot simulate reasonably realistic genomic sequences and patterns!

Having a convincing hypothesis for the origin of the gap, we can devise a strategy for bridging it. In order to build a simulator that both includes a genotype-to-phenotype mapping *and* generates realistic genomic patterns, we need a genotype-to-phenotype mapping that takes as input a realistic genomic sequence. This way, mutations would be performed on realistic sequences (as in bioinformatic simulators) but their effect on fitness would be indirect, depending on their phenotypic effect (as in artificial life simulators).

[2] https://alife.org/encyclopedia/category/software-platforms

The phenotype however, is not required to be realistic. In the next part of this article, we will start from Aevol, an artificial life simulator designed to model the structure of the genotype-to-phenotype map, and extend it in order to evolve biologically realistic sequences. We will then show how the genomic patterns that evolved in this new model can be analyzed by off-the-shelf bioinformatic software.

The Aevol Platform

Aevol is an *in silico* experimental evolution platform specifically designed to study the evolution of genomic structures (Knibbe et al., 2007; Liard et al., 2020; Banse et al., 2023). It is an individual-based model built upon the core idea that the organisation of the genotype-to-phenotype map determines the structure of information coding on the chromosome and hence the way it can be modified by mutational operators. In Aevol, a population of individuals, each owning a double-stranded binary genome, evolves thanks to three processes:

A genome decoding process. The decoding process allows to compute a phenotype and a fitness from the individual's genome. This genotype-to-phenotype map follows the main steps of information decoding in real organisms. Although Aevol uses a binary genome, this makes this platform a perfect starting point to design a model able to evolve realistic sequences. Since the genome decoding process is at the heart of the transformation of Aevol into Aevol_4b, it is described in further details in Figure 1.

A replication process. At each generation, all the individuals compete in order to populate the next generation. The replication scheme is a Wright-Fisher generational scheme and the competition is based on the fitness value calculated from the individual's genome.

A mutational process. At each replication, individuals may undergo mutations. Aevol implements a large set of mutational operators including point mutations, small insertions and deletions and large chromosomal rearrangements (segmental duplications and deletions, inversions, translocations). These operators allow the genome to evolve in sequence but also in size and structure (Banse et al., 2023). This property allows studying how genome length and structure evolve depending on the properties of the population, of the replication and of the mutational processes – typically population size and mutation rates and biases (Rutten et al., 2019; Luiselli et al., 2024).

In Aevol these three processes interact and globally shape the fitness landscape that the population explores during its evolution. Due to the interactions between the mutational process and the genome decoding process, individuals can evolve very different genetic structures from simplistic ones to very complex ones (Liard et al., 2020). Moreover, the fitness landscape is highly irregular, including smooth regions (typically for simple organisms) and rugged ones. Hence, the model can help understand how evolution balances direct selection (*i.e.* selection for fitness) and indirect selection (*i.e.* selection for robustness and evolvability) in various conditions (Knibbe et al., 2007; Banse et al., 2023). Because they are binary, genomes evolved with Aevol cannot be directly compared to those of real organisms. As a consequence, apart from specific situations where the binary nature of the sequences was not an issue (Parsons et al., 2011; Biller et al., 2016), Aevol has remained disconnected from classical bioinformatics and comparative genomics tools, as all other artificial life platforms.

From Aevol to Aevol_4b

The original version of Aevol benefited from the binarity of the genetic sequence as it enabled to keep the genotype-to-phenotype simple and elegant. As shown by Figure 1, three base long codons allow for six amino-acids (AA) to be encoded in addition to the START and STOP codons. These in turn correspond exactly to 3×2 subsequences, hence to three binary codes, themselves corresponding to the three parameters of the triangle-proteins. However, ever since the first versions of the Aevol model over 15 years ago (Knibbe, 2006), the Aevol development team has obviously had in mind the idea of moving to a more realistic 4-bases DNA sequence and several prototypes have been designed over the years (Liard et al., 2017). But none has proved usable, mainly because switching from six AA to twenty AA represents a sharp increase in complexity. Indeed, using a similar decoding process with 20 in place of 6 AA would mean representing the proteins by complex mathematical functions with $\frac{20}{2} = 10$ parameters. All our attempts to encode phenotypes by a linear combination of such complex functions have proven impossible to manage in practice.

Until recently, the situation thus appeared to be deadlocked, and Aevol seemed destined to retain a binary genome. However, a solution emerged that allows increasing the complexity of the model in the sequence realm without increasing the complexity of the functional levels. All that's needed is for the folding algorithm (see Figure 1) to be based on numbering systems using numbering bases greater than two. Indeed, if the encoding of m, w and h are respectively using numbering bases B_m, B_w and B_h, then, a total of $B_m + B_w + B_h$ amino-acids could be used. Hence, providing $B_m + B_w + B_h = 20$, a four bases DNA sequence decoded with the standard genetic code (Figure 2) could be used with only marginal changes of the initial model.

Leveraging this astonishingly simple modification of the model, we choose to encode m in base $B_m = 7$ (AA M0 to M6), w in base $B_w = 6$ (AA W0 to W5) and h in base $B_h = 7$ (AA H0 to H6). These $7+6+7 = 20$ Amino-Acids were randomly put in correspondence with the 20 biological Amino-Acids (Figure 2, outer-ring) such that the standard genetic code could be used in the translation process (Fig-

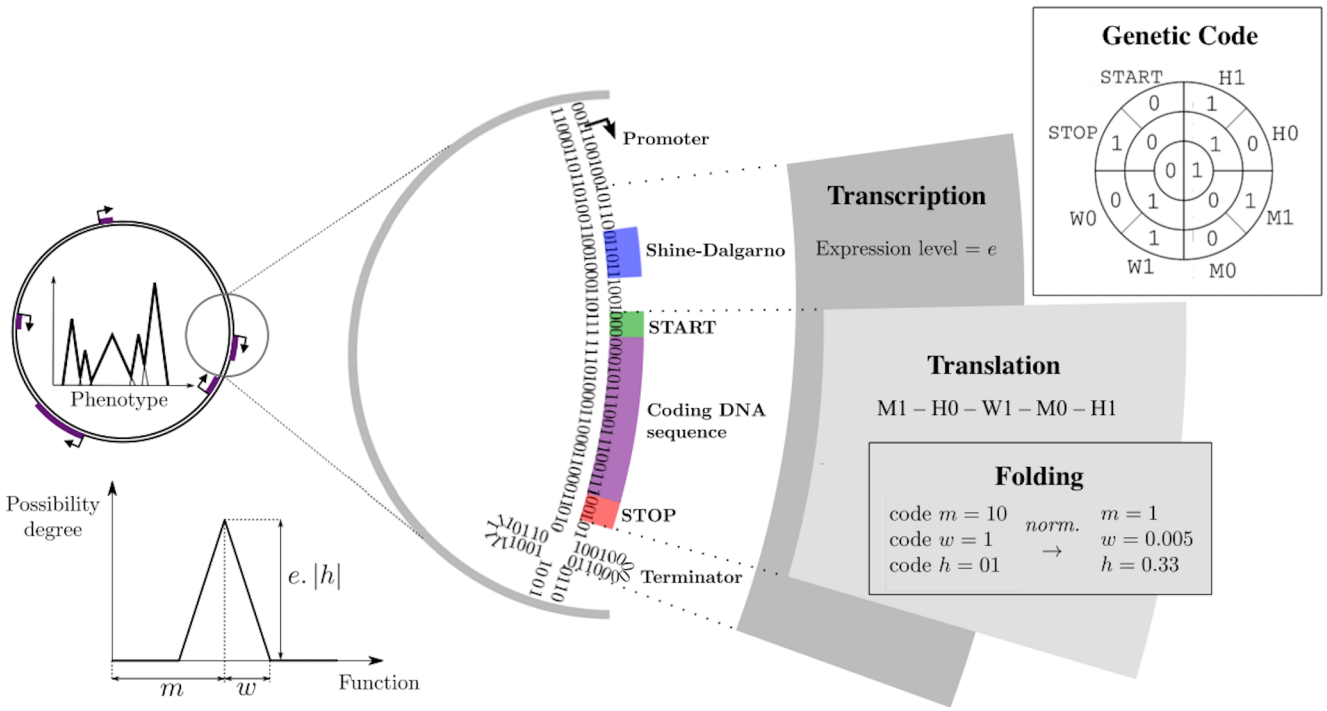

Figure 1: **Genome decoding process in Aevol.** In Aevol the genome is a binary double-stranded sequence on which genes are encoded on both strands (top left). In the model each gene is decoded into a mathematical triangular function (bottom left) through a multistep decoding process including transcription, translation and protein folding. A full description of the model is available at `http://www.aevol.fr`. The central part of the figure displays the decoding of a specific gene through the three following steps:

Transcription: The transcription process searches for promoter sequences on both genome strands. Promoter sequences are 22 base-pair consensus sequences in which up to 4 mismatches are allowed, the number of mismatches determining the expression level e (the more mismatches, the lower the expression level). Once a promoter is found on the sequence, the downstream sequence is transcribed into an mRNA until a terminator is found. Terminators are sequences able to form a hairpin loop akin to ρ-independent prokaryotic terminators (*i.e.* sequences of the form $abcd***\overline{d}\overline{c}\overline{b}\overline{a}$, where $*$ represents any base and x and $\overline{x}$ are complementary base-pairs).

Translation: Aevol searches for genes on each mRNA. A gene is a subsequence starting with a *Shine-Dalgarno* consensus motif (011011) followed, four bases downstream by a START codon (000). The sequence is then read three bases at a time until a STOP codon (001) is found on the same reading frame. The sequence lying between the START and the STOP is a gene. It is translated into a sequence of Amino-Acids (AA) through an arbitrary genetic code (top right). Since codons are three-bases long, there exist 8 codons among which two are reserved for the START and the STOP. The six remaining codons correspond to six AA named M0, M1, W0, W1, H0 and H1. The output of the translation process is the primary sequence of the protein. It is an ordered list of AA corresponding to the ordered sequence of codons in the gene. Here the displayed gene sequence is 000.101.110.011.100.111. Hence, it encodes a protein whose primary sequence is M1-H0-W1-M0-H1 (redundant START codons being ignored).

Protein folding: In Aevol, the protein folding process marks the transition between the sequence realm (DNA, mRNA, protein primary sequences) and the mathematical realm in which the function of the proteins, the phenotypes and the phenotypic target are defined. The folding process first extracts three subsequences from the protein primary sequence: the M, W, and H subsequences, composed respectively of AA M0/M1, W0/W1 and H0/H1. These three subsequences are then transformed into three binary sequences, which are in turn transformed into integers and normalized depending on their lengths with m normalized between 0 and 1, h normalized between -1 and 1 (negative values corresponding to function inhibition) and w normalized between 0 and w_{max} (w_{max} being a parameter setting the maximal pleiotropy level in the model). The folding process outputs three real values in these ranges, that are used as parameters for the triangle-protein function (m corresponding to the mean value, w the the half-width and h to the height of the triangle – notice that h can be negative and that the height of the triangle is weighted by e the expression level of the mRNA on which the gene is located).

Phenotype and fitness computation: Once all the triangle-protein functions (corresponding to all the genes found on all the mRNAs of a given genome) have been computed, they are summed to compute the phenotype (top left). The phenotype is then compared to a reference function called the phenotypic target to compute the individual's fitness (the closer the phenotype to the reference function, the higher the fitness).

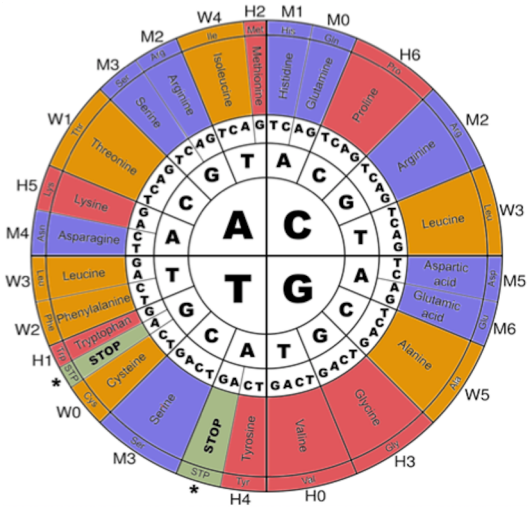

Figure 2: The genetic code and its correspondence with M, W and H digits in Aevol_4b. Notice the three STOP codons (TGA, TAG and TAA). As in most biological organisms, the START codon (ATG) corresponds to the Amino-Acid Methionine (Met). Colors correspond to the amino-acids contributing to a same parameter (m, w or h).

ure 2, inner-rings); Aevol_4b was born!

Although the main structure of the Aevol model is conserved in Aevol_4b, the switch to a 4-bases DNA sequence and the use of the canonical genetic code gives Aevol_4b new properties that allow for direct usage of comparative genomics tools on the simulated sequences. In particular, as shown by Figure 2, the genetic code used in the model is degenerated, the third position of the codon being often redundant. This enables leveraging classical bioinformatic measures such as the dN/dS ratio, hence allowing to estimate parameters such as the intensity of selection or the effective population size directly from Aevol_4b sequences comparison, exactly like bioinformaticians do when analyzing real genomes.

Experimental design

To test Aevol_4b, we simulated a species tree with 99 species diverging from a common ancestor. The 99 final sequences were then provided to bioinformatics researchers who used them to reconstruct the phylogenetic tree using off-the-shelf tools. This "double-blind" test, in which ALife researchers simulate evolution along a tree and bioinformatics researchers analyze the final sequences to reconstruct the tree before comparing the outcome to the simulated ground truth, offers an invaluable way to cross-validate the simulator and the phylogenetic tree inference methods.

Starting from an ancestral population that pre-evolved in Aevol_4b for 2.5 million generations, we generated a random branching tree using the Gillespie SSA Algorithm with a first branching at generation 2,500,000 and a subsequent constant probability of branching of 0.0035 per thousand generations for each branch. The total length of the tree was set to 1 million generations, resulting in a total of 98 branching events and 99 final species. Figure 3 shows the simulated species tree labelled by branching order.

At each branching event, we cloned the whole population and simulated independent evolutions along the resulting two branches but without changing the evolutionary parameters (mutation rates, population size, phenotypic target...) so that they are constant throughout the tree. Finally, we extracted the best individual from each final population and provided the corresponding 99 sequences to the bioinformatics team for independent blind analysis. These sequences are about 30 kbp long and contain an average of 175 genes each, with an average gene length of around 85 bp.

We tested several phylogenomic inference pipelines on this dataset. Two will be presented here: the `progressiveMauve/IQ-TREE` and the `kmer-db/BioNJ` pipelines. To compare the inferred and original trees, we used two metrics: the Robinson-Foulds (RF) metric (Robinson and Foulds, 1981), which compares tree topologies, and the Kuhner-Felsenstein (KF) distance (Kuhner and Felsenstein, 1994), which compares both topologies and branch lengths. For both metrics, lower values indicate better accuracy of the inferred tree. Finally, since inference methods return branch lengths scaled in expected numbers of substitutions per site, we rescaled the tree to match the Aevol branch-length unit (thousands of generations).

Results

`ProgressiveMauve/IQ-TREE` pipeline: This inference pipeline uses a Multiple Sequence Alignment (MSA) tool, `progressiveMauve` (Darling et al., 2009), followed by `IQ-TREE2`, a phylogenomic software that uses the MSA to infer the phylogenetic tree (Minh et al., 2020).

Overall, the more recent clades and closely related species are well inferred (Figure 4). Ancient branches, being more difficult to retrieve, are less accurately inferred. Figure 6 shows that the correctly inferred internal branches and leaves have a length distribution that matches the true one rather well. There are however some exceptions, among which two outliers (indistinguishable on the figure) around $(250, 1750)$ correspond to the two long branches of the inferred tree (Figure 4). Due to the greater heterogeneity of the inferred tree in terms of branch lengths, and because the tree was rescaled to match the true lengths, the numerous smaller branches appear below the $x = y$ axis while the longest branches absorb much of the usable length and appear far above that axis.

The KF distance between this inferred tree and the true one is $23,564$, which is 42.8% of the worst possible value (equal to twice the sum of the original tree branch lengths).

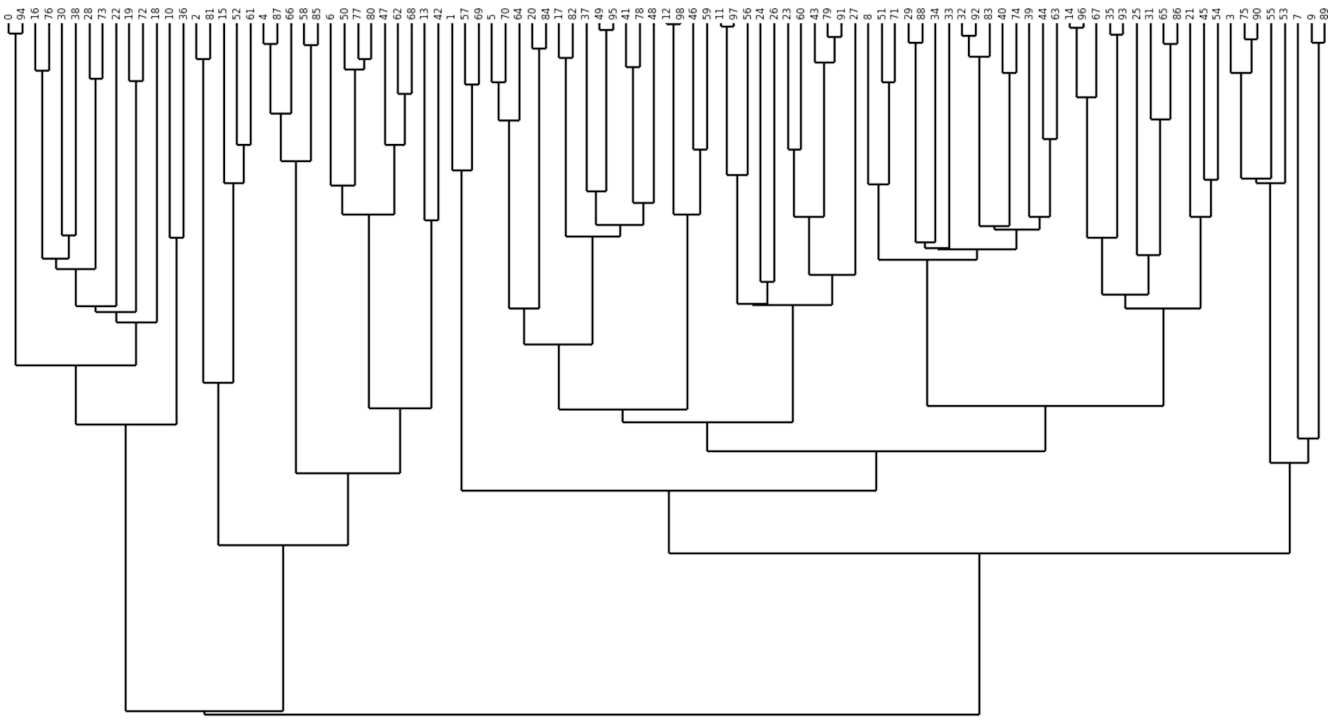

Figure 3: The 99 species tree generated with Aevol_4b

This distance is partly driven by three branches whose lengths are poorly estimated (the two outliers mentioned above and the very short branch 7). The trees have 71 common bipartitions and 25 different ones, resulting in an RF score of $2 \times 25 = 50$ (26% of the worst possible score).

`kmer-db/BIONJ` pipeline: The second inference pipeline is somewhat less sophisticated than the first one. It first uses `kmer-db` (Deorowicz et al., 2019), which calculates a pair-wise distance matrix between all pairs of sequences based on the number of k-mers they share (k-mers being substrings of length k extracted from the DNA sequences). We used the default presets, which sets $k = 18$. We then used `BIONJ` (Gascuel, 1997) for the tree reconstruction, which is an improved version of the neighbor-joining algorithm (Saitou and Nei, 1987).

This pipeline turned out to be very accurate, with an RF error rate of only 8% (*i.e.* 16 wrong branches out of 192 total internal branches when both trees are considered) and a KF error rate of 38% (Figure 5). The lengths of the correctly inferred internal branches and leaves rather match the true lengths, although the internal branches tend to be shorter and the leaves longer than the true values, as shown in Figure 7 and also empirically visible in Figure 5. Moreover, it is worth noting that all wrongly inferred branches are very short branches (in red in Figure 5).

Conclusion and Perspectives

We proposed Aevol_4b, an artificial-life grounded simulator capable of evolving realistic genomic sequences. Using this model, we generated a large species tree and showed that the tree can be recovered, at least partly, by two different bioinformatic pipelines without modifying them in any way. To the best of our knowledge, this is the first successful coupling of artificial life simulations with bioinformatic analysis, and we believe this result opens up significant opportunities for future exchanges between these two fields.

Our results show that the simulated tree can be retrieved from the final sequences, although with varying accuracy for the two tested pipelines. Surprisingly, the simpler pipeline provided the best results, while the classical MSA-based pipeline struggled to infer old branches. This is surprising as the results obtained with the distance-based pipeline indicate that the information is indeed available in the sequences. We are now trying to unravel the origin of this discrepancy.

Before turning to the perspectives opened up by our work, there are two important points that we have deliberately left out throughout the paper, and which need to be discussed.

First, throughout this article, we have used the term "realistic" to describe the genetic sequences generated by Aevol_4b. However, realistic is an intrinsically subjective qualifier that can encompass very different properties depending on the viewpoint. For instance, Yelmen et al. (2021) describe a generative neural network able to produce "high quality realistic genomes" that have little in common with

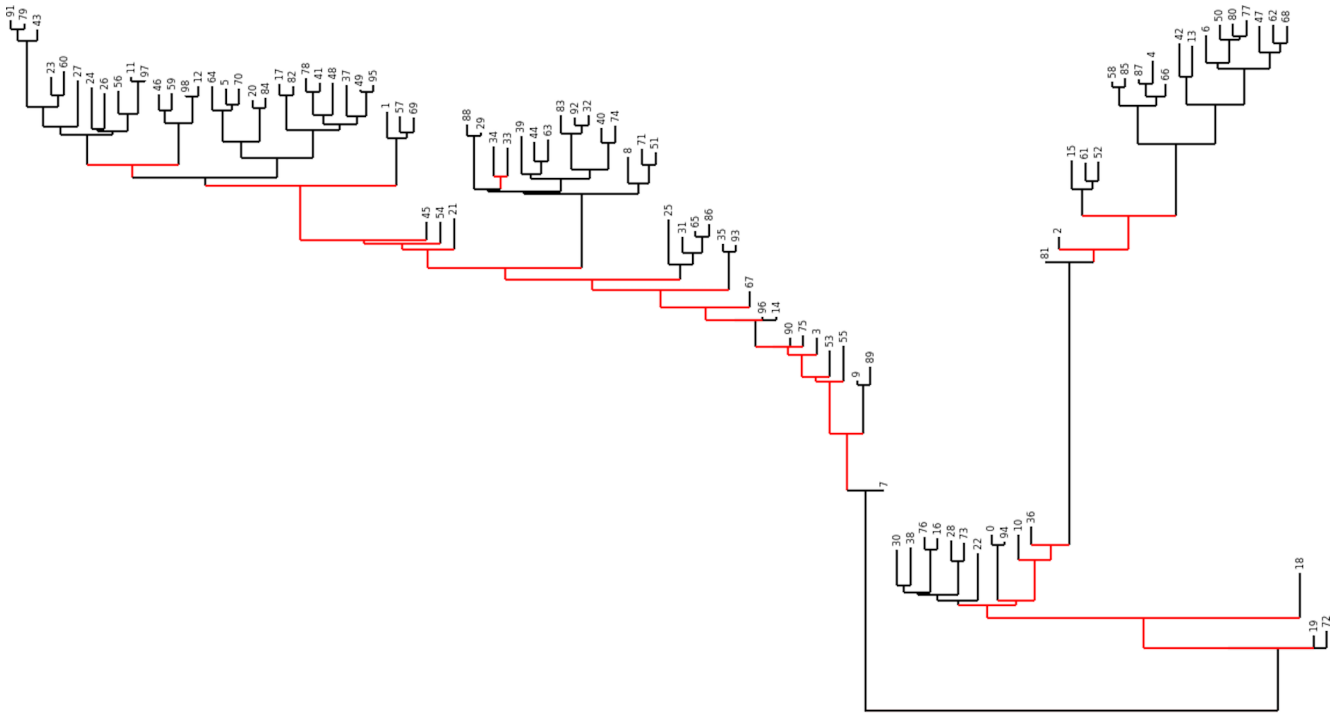

Figure 4: Inferred tree using the `progressiveMauve/IQ-TREE` pipeline. The wrongly inferred branches are colored in red. Although the shape of the tree may look rather different from the original tree (Figure 3), we can see that almost all recent branches are correctly inferred, resulting in scores of RF = 50 (26.0%) and KF = 23,564 (42.8%)

.

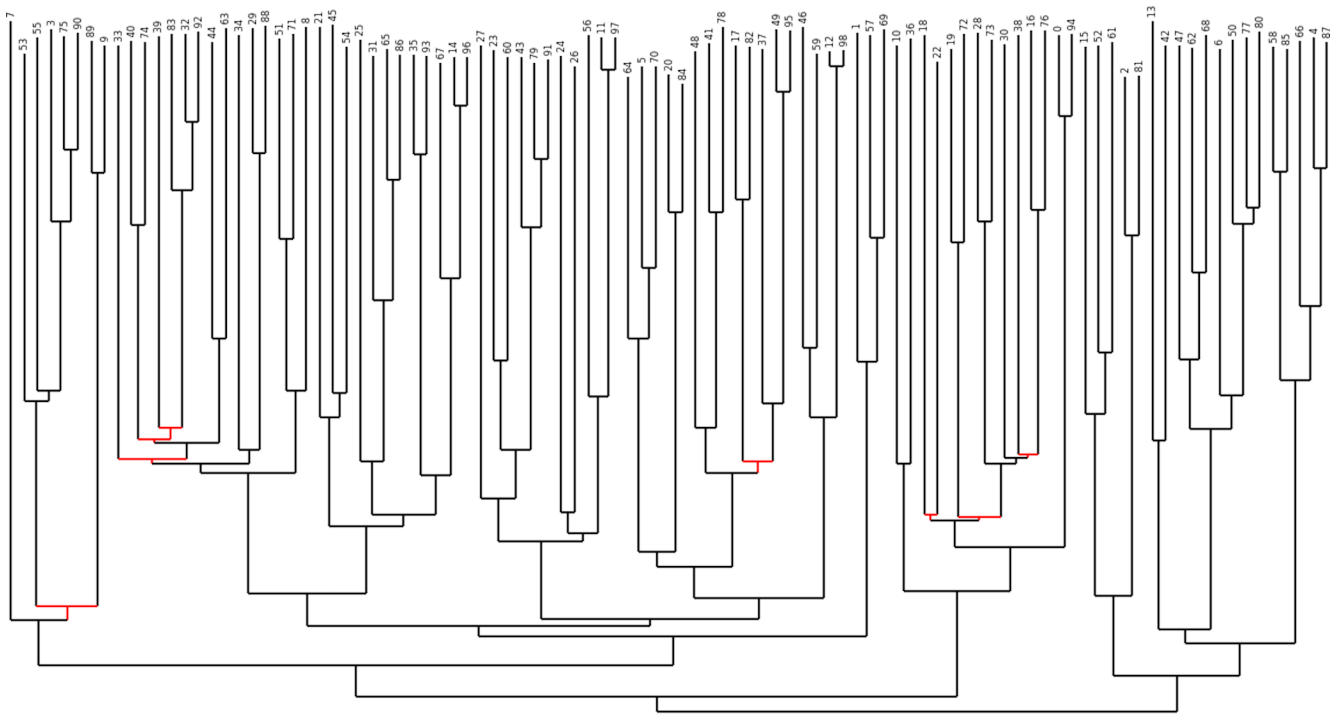

Figure 5: Inferred tree using the `kmer-db/BIONJ` pipeline. The wrongly inferred branches are colored in red. RF = 16 (8.3%); KF = 21,229 (38.6%)

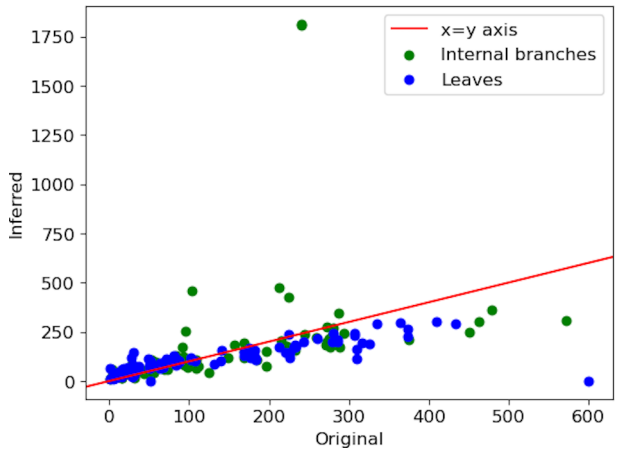

Figure 6: Branch length differences between the correctly inferred branches and the true ones, for the `progressiveMauve/IQ-TREE` pipeline.

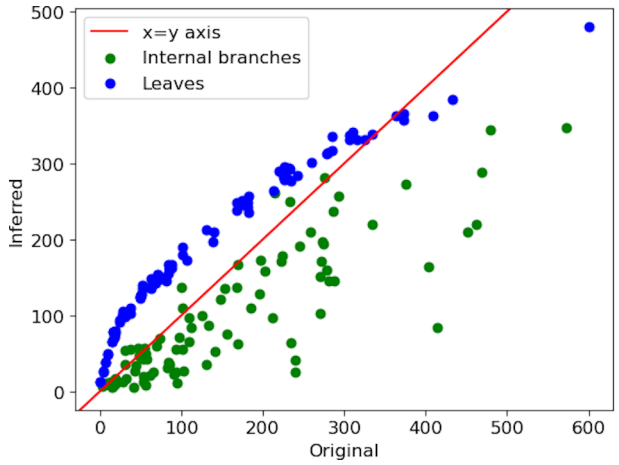

Figure 7: Branch length difference between the correctly inferred branches and the true ones, for the `kmer-db/BIONJ` pipeline.

the genomes produced by Aevol_4b. Indeed, our genomes are only a few tens of kb long and the genes they contain are much shorter than real genes. Here, "realistic" is to be understood relatively to the evolutionary process, meaning that these sequences are the product of evolution and that they encompass enough information for the evolutionary process to be retrieved *a posteriori* – at least by the `kmer-db/BIONJ` pipeline.

Secondly, one might ask why we care about bridging the gap between bioinformatics and artificial life. We strongly believe that both areas could benefit from such bridges. On the bioinformatics side, artificial life could produce independent datasets that would serve as benchmarks to test inference algorithms and to reveal possible caveats. On the artificial life side, bridging the gap allows for analyzing the evolutionary dynamics at work in the simulations using the same measures evolutionary biologists employ when observing evolution. This will enable simulated and real data to be compared quantitatively, whereas to date, artificial life has primarily relied on qualitative comparisons.

Both directions are obvious perspectives of our work. In particular, an immediate perspective would be to generate more complex benchmarks, with irregular trees (*e.g.* including extinction events). We also envision generating species trees in which life traits would change at branching events or along branches. Aevol_4b indeed includes the possibility to modify the mutation rates, population size or environmental conditions. This would allow for testing phylogenetic inference methods on more complex datasets and to evaluate their ability to identify events such as population bottlenecks or environmental drift. Also, while we have tested phylogenetic inference here, other bioinformatic tools and methods could easily be tested, such as the detection of open reading frames, estimation of the nature and intensity of selection, or estimation of the effective population size.

Finally, although we consider Aevol_4b as a decisive step in bridging ALife and bioinformatics, it must be regarded as merely a first step, and the model could be improved in many ways. In particular, Aevol_4b does not include Horizontal Gene Transfer (HGT). Including HGT would allow for simulating much more complex phylogenies in which different parts of the same genome have different evolutionary histories. A similar idea would be to incorporate substitution models such as the General Time Reversible (GTR) model (Tavaré, 1986). At another level, neither Aevol nor Aevol_4b include a speciation model. Adding an ecological level would allow the simulator to generate its own species tree rather than following one provided as an input. Lastly, many AI tools have been recently developed that can predict protein structure accurately (Jumper et al., 2021). We envision leveraging such tools in the genome decoding process. This would allow for evolving more realistic gene sequences, bringing the model even closer to real-life biology.

Acknowledgements

This work was supported by the French Agence Nationale de la Recherche under grant ANR-19-CE45-0010 (Evoluthon).

Data availability

Aevol and Aevol_4b are available from `http://www.aevol.fr`. The 99 sequences of the dataset as well as the original tree file are available from `http://www.aevol.fr/publications/supporting-data/alife2024/`.

References

Adami, C. (2006). Digital genetics: unravelling the genetic basis of evolution. *Nature Reviews Genetics*, 7(2):109–118.

Banse, P., Luiselli, J., Parsons, D. P., Grohens, T., Foley, M., Trujillo, L., Rouzaud-Cornabas, J., Knibbe, C., and Beslon, G. (2023). Forward-in-time simulation of chromosomal rearrangements: The invisible backbone that sustains long-term adaptation. *Molecular Ecology*, Epub ahead of print.

Biller, P., Knibbe, C., Beslon, G., and Tannier, E. (2016). Comparative genomics on artificial life. In *12th Conference on Computability in Europe, Paris, France*, pages 35–44. Springer.

Crombach, A. and Hogeweg, P. (2008). Evolution of evolvability in gene regulatory networks. *PLoS Comp Biol*, 4(7):e1000112.

Dalquen, D. A., Anisimova, M., Gonnet, G. H., and Dessimoz, C. (2012). Alfa simulation framework for genome evolution. *Mol Biol Evol*, 29(4):1115–1123.

Darling, A. E., Mau, B., and Perna, N. T. (2009). Progressive mauve: Multiple alignment of genomes with gene flux and rearrangement. *arXiv:0910.5780*.

Deorowicz, S., Gudyś, A., Długosz, M., Kokot, M., and Danek, A. (2019). Kmer-db: instant evolutionary distance estimation. *Bioinformatics*, 35(1):133–136.

Edgar, R. C., Asimenos, G., Batzoglou, S., and Sidow, A. (2009). Evolver. *Website http://www. drive5. com/evolver*.

Fisher, R. (1930). *The Genetical Theory of Natural Selection*. Oxford University Press.

Gascuel, O. (1997). Bionj: an improved version of the nj algorithm based on a simple model of sequence data. *Mol Biol Evol*, 14(7):685–695.

Haller, B. C. and Messer, P. W. (2019). Slim 3: forward genetic simulations beyond the wright–fisher model. *Mol Biol Evol*, 36(3):632–637.

Hindré, T., Knibbe, C., Beslon, G., and Schneider, D. (2012). New insights into bacterial adaptation through in vivo and in silico experimental evolution. *Nat. Rev. Microb.*, 10(5):352–365.

Hoban, S., Bertorelle, G., and Gaggiotti, O. E. (2012). Computer simulations: tools for population and evolutionary genetics. *Nat. Rev. Genet.*, 13(2):110–122.

Huerta, M., Downing, G., Haseltine, F., Seto, B., and Liu, Y. (2000). Nih working definition of bioinformatics and computational biology. *National Institute of Health*, page 1.

Jumper, J., Evans, R., Pritzel, A., Green, T., Figurnov, M., Ronneberger, O., Tunyasuvunakool, K., Bates, R., Žídek, A., Potapenko, A., et al. (2021). Highly accurate protein structure prediction with alphafold. *Nature*, 596(7873):583–589.

Kauffman, S. and Levin, S. (1987). Towards a general theory of adaptive walks on rugged landscapes. *J. Theor. Biol.*, 128(1):11–45.

Knibbe, C. (2006). *Structuration des génomes par sélection indirecte de la variabilité mutationnelle: une approche de modélisation et de simulation*. PhD thesis, INSA de Lyon.

Knibbe, C., Coulon, A., Mazet, O., Fayard, J.-M., and Beslon, G. (2007). A long-term evolutionary pressure on the amount of noncoding dna. *Mol Biol Evol*, 24(10):2344–2353.

Kuhner, M. K. and Felsenstein, J. (1994). A simulation comparison of phylogeny algorithms under equal and unequal evolutionary rates. *Mol Biol Evol*, 11(3):459–468.

Langton, C. (1988). Artificial lifepreface. *Artificial Life. Addison Wesley, Reading MA, pp. I–IX*.

Liard, V., Parsons, D. P., Rouzaud-Cornabas, J., and Beslon, G. (2020). The complexity ratchet: Stronger than selection, stronger than evolvability, weaker than robustness. *Artificial life*, 26(1):38–57.

Liard, V., Rouzaud-Cornabas, J., Comte, N., and Beslon, G. (2017). A 4-base model for the aevol in-silico experimental evolution platform. In *Artificial Life Conference Proceedings*, pages 265–266. MIT Press, Cambridge, MA.

Luiselli, J., Rouzaud-Cornabas, J., Lartillot, N., and Beslon, G. (2024). Genome streamlining: effect of mutation rate and population size on genome size reduction. *bioRxiv*, 2024.03.14.584996.

Mallo, D., de Oliveira Martins, L., and Posada, D. (2016). Simphy: phylogenomic simulation of gene, locus, and species trees. *Systematic biology*, 65(2):334–344.

Minh, B. Q., Schmidt, H. A., Chernomor, O., Schrempf, D., Woodhams, M. D., von Haeseler, A., and Lanfear, R. (2020). IQ-TREE 2: New Models and Efficient Methods for Phylogenetic Inference in the Genomic Era. *Mol Biol Evol*, 37(5):1530–1534.

Parsons, D. P., Knibbe, C., and Beslon, G. (2011). Homologous and nonhomologous rearrangements: Interactions and effects on evolvability. In *Proc. ECAL 11*, pages 622–629.

Peng, B., Chen, H.-S., Mechanic, L. E., Racine, B., Clarke, J., Clarke, L., Gillanders, E., and Feuer, E. J. (2013). Genetic simulation resources: a website for the registration and discovery of genetic data simulators. *Bioinformatics*, 29(8):1101–1102.

Robinson, D. and Foulds, L. (1981). Comparison of phylogenetic trees. *Mathematical Biosciences*, 53(1):131–147.

Rutten, J. P., Hogeweg, P., and Beslon, G. (2019). Adapting the engine to the fuel: mutator populations can reduce the mutational load by reorganizing their genome structure. *BMC evolutionary biology*, 19:1–17.

Saitou, N. and Nei, M. (1987). The neighbor-joining method: a new method for reconstructing phylogenetic trees. *Mol Biol Evol*, 4(4):406–425.

Tavaré, S. (1986). Some probabilistic and statistical problems on the analysis of dna sequence. *Lecture of Mathematics for Life Science*, 17:57.

Trost, J., Haag, J., Höhler, D., Jacob, L., Stamatakis, A., and Boussau, B. (2024). Simulations of sequence evolution: how (un) realistic they are and why. *Mol Biol Evol*, 41(1):msad277.

Wilke, C. O., Wang, J. L., Ofria, C., Lenski, R. E., and Adami, C. (2001). Evolution of digital organisms at high mutation rates leads to survival of the flattest. *Nature*, 412(6844):331–333.

Yelmen, B., Decelle, A., Ongaro, L., Marnetto, D., Tallec, C., Montinaro, F., Furtlehner, C., Pagani, L., and Jay, F. (2021). Creating artificial human genomes using generative neural networks. *PLoS genetics*, 17(2):e1009303.