Dynamics of genetic code evolution: The emergence of universality

John-Antonio Argyriadis, Yang-Hui He, Vishnu Jejjala, and Djordje Minic

Phys. Rev. E **103**, 052409 — Published 17 May 2021

DOI: 10.1103/PhysRevE.103.052409

Dynamics of genetic code evolution: The emergence of universality *

John-Antonio Argyriadis[a], Yang-Hui He[b], Vishnu Jejjala[c], Djordje Minic[d]

[a] *Jesus College, University of Oxford, OX1 3DW, UK;*
Rudolf Peierls Centre for Theoretical Physics, Clarendon Laboratory, Parks Rd, University of Oxford, OX1 3PU, UK

[b] *Department of Mathematics, City, University of London, EC1V 0HB, UK;*
Merton College, University of Oxford, OX1 4JD, UK;
School of Physics, NanKai University, Tianjin, 300071, P.R. China

[c] *Mandelstam Institute for Theoretical Physics, School of Physics, NITheP, and CoE-MaSS, University of the Witwatersrand, Johannesburg, WITS 2050, South Africa;*
David Rittenhouse Laboratory, University of Pennsylvania, Philadelphia, PA 19104, USA

[d] *Department of Physics and Center for Soft Matter and Biological Physics, Virginia Tech, Blacksburg, VA 24061, USA*

E-mail: john-antonio.argyriadis@jesus.ox.ac.uk, hey@maths.ox.ac.uk, vishnu@neo.phys.wits.ac.za, dminic@vt.edu

Abstract: We study the dynamics of genetic code evolution. The model of Vetsigian et al. [1] and Vetsigian [2] uses the mechanism of horizontal gene transfer to demonstrate convergence of the genetic code to a near universal solution. We reproduce and analyze the algorithm as a dynamical system. All the parameters used in the model are varied to assess their impact on convergence and optimality score.We show that by allowing specific parameters to vary with time, the solution exhibits attractor dynamics. Finally, we study automorphisms of the genetic code arising due to this model. We use this to examine the scaling of the solutions in order to re-examine universality and find that there is a direct link to mutation rate.

* The author list is alphabetical.

Contents

1 Introduction

The genetic code arose through evolution. We think of it as being universal, optimal, and highly redundant [3]. The mechanics of the evolution of life on Earth means that all organisms share the same genetic code in which each codon produces the same amino acid. We call this the **standard genetic code**. (While there are deviations from the standard genetic code [4], these involve only a handful of codons being coded differently and will not be the subject of our investigations here.) Because of selection pressures, the standard genetic code has self optimized to minimize errors in translation and transcription [5–7]. From theoretical considerations, Woese et al. [8] showed that the standard genetic code is related to a property called the *polar requirement*, which has subsequently been corroborated by experiment [9] and shown to be highly optimal when considering one type of error: point mutations [10]. It can be considered as part of an abstract chemical property of the genetic code [11]. Following Vetsigian et al. [1] and Vetsigian [2], in this paper, we consider the code's optimality in terms of these features.

Sella and Ardell [11] attempted to model the evolution of the genetic code. This was done through considering the coevolution between the genetic code and the encoding of a protein within a closed model system. This allows for complex dynamics between mutations of messages and selection on proteins in order to minimize the lethal effects of these mutations. This minimizes the errors through mutations and allows protein networks to develop to promote a higher likelihood of survival. We seek to understand whether the algorithm based on this coevolutionary model can be phrased as a purely physical problem of dynamical evolution. To address this we must first discuss the algorithm.

The algorithm described by Vetsigian et al. [1] models the evolution of the genetic code through **horizontal gene transfer (HGT).** This allows organisms to exchange genetic information via DNA through transferring the segments of a genome to each other within the same generation through various mechanisms. This is used in concert with the Code Message Coevolution Model dynamics described by Sella and Ardell [11] to obtain an iterative discrete time algorithm. Implementation of the algorithm demonstrates convergence of the genetic code to a highly optimised and near universal solution. This solution is an attractor. Horizontal gene transfer is crucial to achieving this solution. This result provides a testable model for understanding the standard genetic code.

The dynamics from the coevolution model along with the additional communi-

cation incorporated by the iterative discrete time algorithm governs the evolution of genetic code states (genetic code configurations) of which there is a finite number. This allows us to treat the algorithm as a discrete dynamical system, as it is the time dependent dynamics of coevolving matrices which represent the genetic code state. We will consider this algorithm as if it were a system out of equilibrium for which there is the emergence of an attractor solution in the space of genetic code mappings [12].

The behavior we establish exemplifies the notion of *universality*. In statistical physics, systems with a large number of degrees of freedom exhibit universality in a scaling limit. Historically, this idea originated in the theory of phase transitions and was made mathematical precise through the renormalization group. (See, for example, [13–15].) Starting from specific initial conditions, by integrating out degrees of freedom (*e.g.*, through coarse-graining), we flow to a fixed point. At the fixed point, the theory is scale invariant.[†] Very different physical systems can flow to the same fixed point in that correlation functions of local operators behave in an identical manner. Such theories are said to belong to the same universality class. Input parameters dictate how the system converges to universality. It is therefore natural to examine how variations on these parameters influence the universality of the solution. We will use approximate scale invariance as a tool to assess how close we are to universality and to diagnose features of the universal solution.

In this article, we investigate a mechanism for the origin of the genetic code that leads to universal behavior at late times. We initially describe the model with considerations of universality as in Vetsigian et al. [1] by defining it in such a way that all entities within the algorithm converge to a single solution. We then consider universality in a more formal manner through statistical mechanics.

Let us begin by stating the main results. The first requirement for the convergence of the genetic code is a trivial observation that we make rigorous: there must be more codons than amino acids. The second requirement is that we must demand horizontal gene transfer to optimize the setup. This corroborates the claims of Vetsigian et al. [1]. We also discover that universality in terms of scaling in the solution depends on the rate of mutations. It is largely independent of mistranslations of the genetic code.

The organization of this paper is as follows. In Section 2, we recast the biological algorithm into a computational algorithm to which we can apply the principles of dynamical systems. We reproduce the results from Vetsigian et al. [1] and show that the initial conditions do not influence the algorithm's ability to flow to a near universal

[†] This is a simplification. At a fixed point of the renormalization group, the theory enjoys a larger symmetry, *conformal invariance*, which includes scale or dilatation invariance.

solution. Our new results are in Section 3. In particular, in Section 3.1, we correct minor errors in the literature. In Sections 3.2–3.4, we vary all the parameters in the model in order to examine their influence on the attractor mechanism. In Section 3.5, we discuss automorphisms and scaling in the genetic code. We then illustrate the mechanism for universality in terms of scaling with an example. In Section 3.6, we re-examine universality and express this behavior in terms of the homogeneity of the fitness function. We find that the universal solution is characterized by the rate of mutations and is largely independent of the mistranslation rate. In Section 4, we conclude with a summary and directions for future work. Finally, the Appendices collect the results of various experiments less central to the argument than those discussed in the main text.

2 Modeling framework

We begin with a precise rephrasing of the problem addressed by Vetsigian et al. [1] into one of computation. Emphasis will be on representing the aspects of the mathematical problem while minimizing the amount of biology introduced.

2.1 Basic definitions

We model the set of codons making up DNA geometrically using a Hamming metric [16].

DEFINITION 1. *Let $i_b \in i$ be a set of elements (**bases**) forming an alphabet of length $|i|$. We define a **codon** as a sequence of n bases such that $c \in \mathcal{C} := \{i_{b,1}, \ldots, i_{b',n}\}$. The number of possible codons is $|\mathcal{C}| = |i|^n$.*

For codons in the standard genetic code, we have $|i| = 4$ (A,C,G,T) and $n = 3$ meaning that $|\mathcal{C}| = 64$. We define the set of codons, $\mathcal{C}$, lexicographically. Note that there is an associated symmetry with a Hamming metric [17].

We can next define the structure of the genetic code:

DEFINITION 2. *Denoting the set of animo acids $a \in \mathcal{A}$ such that we have $|\mathcal{A}|$ amino acids, the mapping from codon space to amino acid space, $G : \mathcal{C} \to \mathcal{A}$ is the **genetic code**. We represent the map G as matrix $\Delta_{c,a}$ with dimensions $|\mathcal{C}| \times |\mathcal{A}|$ such that:*

$$\Delta_{c,a} = \begin{cases} 1 & \text{if } G(c) = a \ , \\ 0 & \text{otherwise} \ . \end{cases} \tag{2.1}$$

We refer to $\Delta_{c,a}$ as the **delta matrix**. This matrix defined in (2.1) has one entry per row (as each codon can only map to a single amino acid) and no empty columns (we assume that every amino acid has been mapped to). The map G is surjective but not injective. In Nature, we encounter 20 amino acids in the genetic code.[‡] The codon TTT maps to the amino acid phenylalanine, for example. This matrix can therefore be considered as a non-square, row-stochastic, binary matrix with no empty columns. Notice that when summing over the columns of a row-stochastic matrix, we get 1. These properties place constraints on the information flow for optimization [18]. Note that due to these constraints, we also must have $|\mathcal{C}| \geq |\mathcal{A}|$. Using the inclusion/exclusion principle [19], the number of possible configurations of the delta matrix $\Delta_{c,a}$ is

$$\#_{\text{config}} = \sum_{j=0}^{|\mathcal{A}|} (-1)^j \binom{|\mathcal{A}|}{j} (|\mathcal{A}| - j)^{|\mathcal{C}|} \ . \tag{2.2}$$

During the map G, errors occur with a given probability:

$$\text{prob}(c \to a) = \sum_{c'} L_{c,c'} \Delta_{c',a} \ , \qquad L_{c,c'}(\ell) = \begin{cases} \frac{\ell}{(n(|i|-1))} & \text{dist}(c,c') = 1 \ , \\ 1-\ell & \text{dist}(c,c') = 0 \ , \\ 0 & \text{otherwise} \ . \end{cases} \tag{2.3}$$

The parameter $\ell \in [0,1] \subset \mathbb{R}$. The distance is defined using the Hamming metric such that:

$$\text{dist}(c,c') := \#(c_j \neq c'_j)_{j=1,\ldots,n} \ . \tag{2.4}$$

The distance is the number of bases i_b that differ between two codons c and c'. Here, ℓ represents a parameter for the probability of error, and $L_{c,c'}(\ell)$ is a bistochastic matrix, *viz.*, a symmetric, non-negative matrix whose rows and columns sum to 1. The matrix $L_{c,c'}(\ell)$ is used in order to only consider nearest neighbors in codon space. The number of codons with $\text{dist}(c,c') = 1$ is given by $n(|i|-1)$ as there are n positions which can have $|i|-1$ different values. We can encode this information in a *Hamming graph* in which the $\mathcal{C} = |i|^n$ possible codons correspond to vertices and an edge joins vertices whose corresponding codons that differ by a single letter — *i.e.*, those codons at Hamming distance 1.

In this algorithm the genetic code G rearranges itself in order to minimize the likelihood that probabilistic nature of the map causes a differing amino acid a to appear when mapping from codon space [6, 7]. In this model we consider two forms of errors, both of which only occur on nearest neighbors ($\text{dist}(c,c') = 1$). They are the following:

[‡] For the purposes of calculation, we treat the stop codons as mapping to a dummy amino acid, so in our language $|\mathcal{A}| = 21$ in the standard genetic code.

1. **Mistranslation**: When a single base i_b is read incorrectly. We will denote this $T_{c,c'}$ and take $\ell \to \nu$, where ν is the rate of mistranslation.

2. **Point mutations**: A single base i_b changes before being read. We will denote this $M_{c,c'}$ and $\ell \to \mu$ where μ is the rate of mutations. There are various kinds of point mutations.

For simplicity, we will neglect excisions or insertions of bases.

2.2 Fitness

Information is translated from genome to proteome. For our purposes, these are sequences of codons and amino acids, respectively. In particular,

DEFINITION 3. *A sequence $S_G = \{c_1, \ldots, c_M\}$ of length M is called a **genome**, where each codon $c_x \in \mathcal{C}$ has a position x in the sequence $\{1, \ldots, M\}$. A target amino acid $s(x)$ is the mapping under the genetic code G of the codon at position x to the amino acid $s(x) \in \mathcal{A}$. The image under the genetic code map G of the genome sequence S_G, gives a sequence $S_P = \{s_1, \ldots, s_M\}$ of target amino acids called the **proteome**, which is a subsequence of a protein.*

We denote the target amino acid $s(x)$ as s in order to abbreviate notation. The definition we quote above is a slight simplification of [1, 2, 11] as in this algorithm, we assume $s \equiv a \in \mathcal{A}$. As the amino acids at each position in the sequence are indistinguishable [1], we can store details of the proteome and genome within the following objects:

DEFINITION 4. *A vector L_s specifies the frequency of the target amino acid s in a proteome sequence of length M. The codon usage matrix $U_{c,s}$ specifies the frequency of a codon c within a target amino acid s.*

Crucially, we encode all necessary information about the genome and proteome within these two objects without having to go through the respective sequences analytically. The two matrices are both column stochastic, *i.e.*,

$$\sum_{s}^{|\mathcal{A}|} L_s = 1 \ , \qquad \sum_{c}^{|\mathcal{C}|} U_{c,s} = 1_s \ . \tag{2.5}$$

The notion of distance in amino acid information space is structurally ambiguous (not well defined like a Hamming metric). Due to this we can define the topological distance between amino acids by the following $a_d \in [0, 1]$, which can be randomly generated. The notion of distance is normalized. Using this we can define a **fitness matrix** as:

$$W_{a,s} = \Phi^{|a_d - s_d|} \ . \tag{2.6}$$

As in Sella and Ardell [11], Φ is a parameter used to consider how an abstract physicochemical distance between amino acids scales into the fitness. This makes the fitness matrix (2.6) some measure of how "useful" each arbitrary amino acid a is instead of the target amino acid s. Since $0 < \Phi \leq 1$, this is a positive symmetric matrix. By considering the probability of mistranslations and the entire genome we can describe an overall fitness score [1]:

$$f = \prod_c \prod_s \{\sum_{c'} \sum_a T_{c,c'} \Delta_{c',a} W_{a,s}\}^{L_s U_{c,s}} \ . \tag{2.7}$$

This product is taken component wise.

To measure how well a delta matrix performs, we define the **optimality score** O as:

$$O = \sum_c \sum_{c'} (N_{c,c'} \{\sum_a \sum_b \Delta_{c,a} S_{a,b} \Delta^T_{c',b}\}) \ , \tag{2.8}$$

which measures the average amino acid similarity between neighboring codons. We define amino acid similarity as $S_{a,b} = \sum_s |W_{a,s} - W_{s,b}|$. In (2.8), $N_{c,c'}$ is 1 if two codons are nearest neighbors ($\text{dist}(c, c') = 1$) and zero otherwise [1, 2]. Note that it is a tautology that two isomorphic genetic codes give the same optimality score. We return to this point in Section 3.5 below.

2.3 The algorithm

Based on these mathematical preliminaries, we consider the following algorithm [1].

1. **Construction:** We can construct a set of N objects each with their own genetic code G and therefore delta matrix $\Delta_{c,a}$ and their own codon usage matrix $U_{c,s}$.

2. **Mixing:** We randomly select one object as the acceptor A and a random subset K of N as the donors ($k \in K \subset N$) and run them through the iteration:

$$(1 - H) U^A_{c,s} + \frac{H}{K} \sum_{k \in K} U^{(k)}_{c,s} \to U^A_{c,s} \ . \tag{2.9}$$

H represents the fraction of the genetic code due to horizontal gene transfer ($H \in [0, 1]$).

3. **Fitness maximization:** We attempt an elementary code change to the delta matrix $\Delta_{c,a}$. We do this by assigning one codon to a new amino acid. This is done by reallocating a unit entry in $\Delta_{c,a}$ to a different position within that row of the matrix. We accept the new code if and only if it preserves or increases the fitness score f, which has been calculated using the new $U^A_{c,s}$. Otherwise, we keep the original delta matrix $\Delta_{c,a}$, if there are no new possibilities.

4. **Mutational equilibrium:** We can derive a new codon usage matrix $U^A_{c,s}$ from the new delta matrix $\Delta_{c,a}$ uniquely at mutational selection equilibrium. We first derive a fitness matrix with respect to codons $F_{c,s} = \sum_a \Delta_{c,a} W_{a,s}$. Using the Perron–Frobenius theorem, we calculate the column stochastic eigenvector corresponding to the largest eigenvalues, for the following matrix (Q^s):

$$Q^s_{c,c'} = \sum_{c''} M_{c,c''} \delta_{c'',c'} F_{c'',s} \ , \tag{2.10}$$

where $\delta_{c'',c'}$ is a Kronecker delta so that we consider the s^{th} column of the matrix $F_{c'',s}$ as a diagonal matrix. The index s here is fixed and not a free index. Each column stochastic eigenvector of $Q^s_{c,c'}$ corresponds to the s^{th} column of $U^A_{c,s}$. We normalize the eigenvector so that it is column stochastic) by setting the sum of elements to unity.

5. **Repetition:** We repeat steps 2 through 4 for t time steps.

Experimental setup: In this model there are 12 parameters to generate and define. These are tabulated as follows.

- **Space structure:** $|i|$ and n for the codon space, $|\mathcal{A}|$ and a_d for amino acid space, and L_s target amino acid frequency;
- **Innovation pool structure:** N number of objects, K number of donors per iteration, and H fraction of genome that is similar due to horizontal gene transfer;
- **Noise and fitness parameters:** ν, μ, and Φ;
- **Number of time steps:** t.

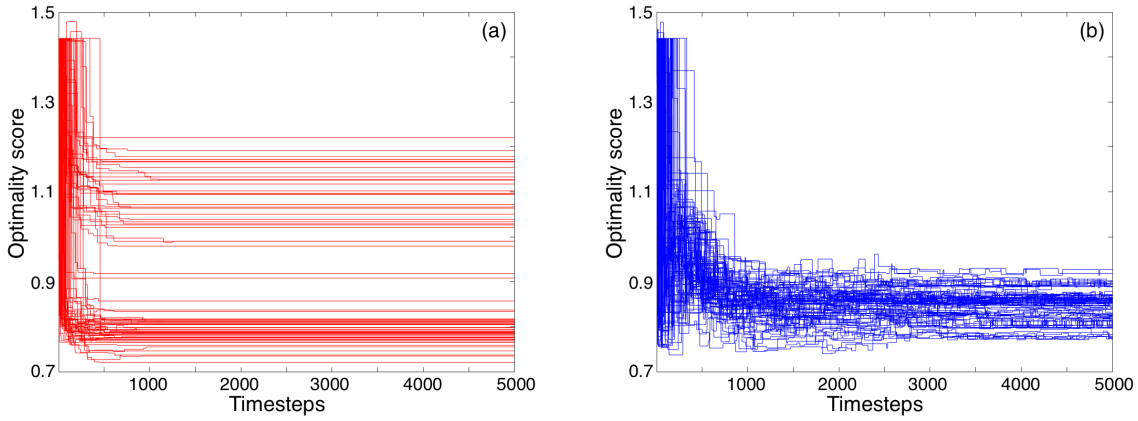


Figure 1: Evolution of optimality score for (a) $H = 0$ (red) and (b) $H = 0.4$ (blue). Both graphs were produced using the following parameters: $|i| = 4$, $n = 3$ giving $|\mathcal{C}| = 64$, $|\mathcal{A}| = 20$, $N = 80$, $K = 1$, $\nu = 0.01$, $\mu = 10^{-4}$, and $\Phi = 0.99$. L_s and a_d are the same for both graphs.

We generate L_s and a_d randomly. The parameters $|i|$, n, $|\mathcal{A}|$, N, and K are positive integers while H, ν, μ, and Φ take any value in the interval $[0, 1]$. We reproduce the results in Vetsigian et al. [1] using their parameters as quoted in Figure 1. The initial delta matrices $\Delta_{c,a}$ are identical at the start, as done in Vetsigian et al. [1].The results in Figure 1 show that when modeling without horizontal gene transfer (Figure 1 (a), red), the delta matrices $\Delta_{c,a}$ optimize themselves, but do not converge to a universal solution. This is shown by the optimality scores, O, ranging from 0.7 to 1.25 and not changing after 1500 time steps. When including horizontal gene transfer (Figure 1 (b), blue) we get a set of optimality scores, O, that optimize on average more than without horizontal gene transfer (red, $H = 0$). The results display the attractor mechanism. This is because the optimality score falls in a smaller range (between 0.75 and 1) and fluctuations continuing at the the $t = 5000$ time step.

The time taken to produce these results was very large as we use $|i| = 4$, $n = 3$, and $|\mathcal{A}| = 20$ giving us 64×20 matrices. To perform a more careful analysis, we consider a toy model by reducing the matrix dimensions to 27×9. This corresponds to the parameters $|i| = 3$, $n = 3$ (so $|\mathcal{C}| = 27$) and $|\mathcal{A}|$=9. We also set up the algorithm so that each entity has its own unique delta matrix $\Delta_{c,a}$ such that they all start with different initial optimality scores in order to see if the scores will still converge. These results are in Appendix A. They show some convergence after 5000 time steps. This suggests that the set of $\Delta_{c,a}$ can be arbitrary in order for optimised attractor mechanism to emerge. This also points to the existence of an attractor mechanism at work for $H \neq 0$.

When performing the analysis all initial delta matrices $\Delta_{c,a}$ will be identical. We will vary a single parameter from the set $\{|i|, n, |\mathcal{A}|, N, K, H, \nu, \mu, \Phi\}$ while keeping the others fixed. We generate a_d and L_s randomly for all runs. We will take ten runs for each parameter and take the average. The standard deviation will be used to analyze the spread (to measure the rate of convergence). We use the standard deviation as it allows us to measure the spread of optimality scores in an intuitive manner, similar to the mean code distance used in Vetsigian et al. [1] as we highlight further in Section 3.5. We will take the average of the standard deviation over the ten runs. These results are discussed in Sections 3.2–3.4 below, where we consider universality to be represented by the ability for all entities within the algorithm to converge to a single solution. The remainder of Section 3 is devoted to re-defining and re-examining universality using statistical mechanics and the renormalisation group.

3 Results and analysis

3.1 The model of Sella and Ardell

In an insightful paper, Sella and Ardell [11] develop a Code Message Coevolution Model that describes the impact of message mutation on the fitness of the genetic code. The authors observe that at mutational equilibrium, there is a balance between mutations in messages and selection on proteins. This model has been summarized in Becich et al. [20]. The process involves calculating the column stochastic eigenvector corresponding to the largest eigenvalues as described in step 4 (mutational equilibrium) of the algorithm described in Section 2.3. We note for completeness and reproducibility two minor errata. Example A from Sella and Ardell [11] consists of a model with the following setup. There is a ring of five codons mapping to a ring of five amino acids with $\Delta_{c,a}$ being the identity matrix. Note that we will denote the eigenvector as $U_{c,s}$, however, s is fixed and is not a free index (as in our step 4). To reproduce the results, we take $\Phi = 0.8^5$ and $\mu = 0.01$.[§] The resulting eigenvalues and eigenvectors are given in Table 1. Note that at machine precision the eigenvectors sum to one ($\sum_c U_{c,s} = 1$) as required.

[§] We are grateful to D. Ardell for communications on this point.

Eigenvalues and column stochastic eigenvectors for a set of Φ and μ

	Scaling for an abstract physicochemical between amino acids Φ	Rate of mutations μ	Largest eigenvalue λ^s
1	0.8	0.1	0.9549
2	0.8^5	0.01	0.9808

	Corresponding eigenvector ${U_{c,s}}^T$
1	$\begin{bmatrix}0.2635 & 0.2134 & 0.1549 & 0.1549 & 0.2134\end{bmatrix}$
2	$\begin{bmatrix}0.9058 & 0.0461 & 0.0011 & 0.0011 & 0.0461\end{bmatrix}$

Table 1: Row **1** gives the resulting eigenvalues and eigenvectors for the parameters stated in the paper. Row **2** provides the eigenvalues and eigenvectors from the paper using the corrected parameters. Note these eigenvectors are for a given value of s.

3.2 Varying parameters for space structure

Recall that $|i|$ counts the number of nucleotides. These are the letters that comprise a DNA sequence. Taking a codon to consist of an n nucleotide sequence, the number of possible codons is then $|\mathcal{C}| = |i|^n$. These codons describe $|\mathcal{A}|$ amino acids. With four bases as in the standard genetic code, there are 64 three base sequences corresponding to possible codons. These codons correspond to 20 amino acids, so we have a many to one map. In this subsection, we report on experiments involving varying parameters corresponding to the spatial structure of the map.

Experiment 1: Varying the number of nucleotides
We test the fitness optimization for the cases $|i| = 3, 4, 5$. We do not take $|i| = 2$ as this gives $|\mathcal{C}| = 8 < |\mathcal{A}| = 9$, breaking one of the constraints on the delta matrix. We do not take $|i| \geq 6$ either as this produces a matrix that is at minimum 216×9, which takes significant processing time to iterate. The results are given in Figure 2. For all values of $|i|$, Figure 2 (a) displays an optimised solution with an attractor mechanism converging to a near universal solution.When varying $|i|$ we find the value of the optimality score, O, increases proportionally as shown in Figure 2 (b). This makes sense as increasing $|i|$ increases $|\mathcal{C}|$ meaning we sum over more elements to get the optimality score O. The rate of convergence decreases as indicated by the error bars increasing proportionally with $|i|$ in Figure 2 (b). This is as expected as larger matrices should take longer to find the universal solution.

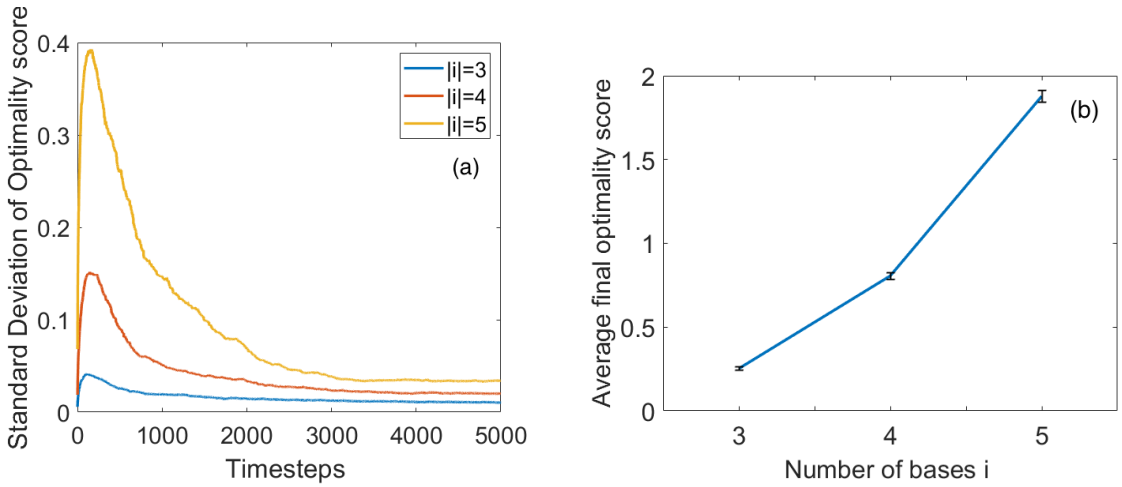


Figure 2: Figure 2 (a) shows the average time evolution of the standard deviation of the optimality score for a given $|i|$ over ten runs. Figure 2 (b) shows the average final optimality score for a given $|i|$ over ten runs. The initial parameters are the same for all runs: $n = 3$, $|\mathcal{A}| = 9$, $N = 80$, $K = 1$, $H = 0.4$, $\nu = 0.01$, $\mu = 10^{-4}$, and $\Phi = 0.99$. The error bars show the average one standard deviation spread of final optimality scores over the ten runs (to measure the rate of convergence).

Experiment 2: Varying the length of a codon

For the length of a codon n, we take $n = 2, 3, 4$. We do not take $n = 1$ or $n \geq 5$ for the same reasons as when varying $|i|$. The results are given in Figure 3. They display the same pattern as when varying $|i|$, because we are increasing $|\mathcal{C}|$ again. When $n = 2$, we get $|\mathcal{C}| = |\mathcal{A}| = 9$ which means $\Delta_{c,a}$ forms a permutation matrix. This matrix cannot be changed in step 3 of the algorithm discussed in Section 2.3 (fitness maximization) as we cannot reassign a single codon c to a new amino acid a without being left with an empty column. The delta matrix, $\Delta_{c,a}$, cannot therefore evolve, giving a single flat line for $n = 2$ as seen in Figure 3 (a). This implies that we require $|\mathcal{C}| > |\mathcal{A}|$ for the algorithm to work. We take note of the smallness of the standard deviations in Figure 3 (b).

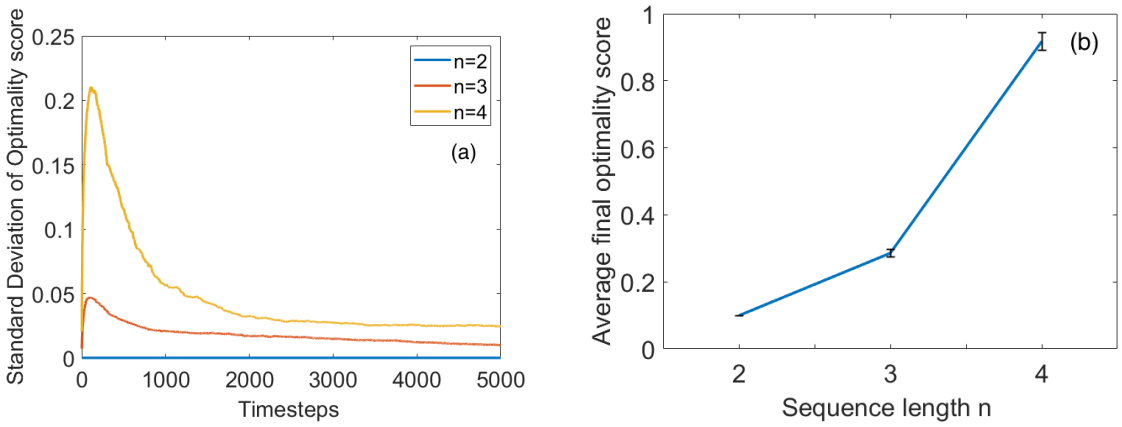


Figure 3: Figure 3 (a) shows the average time evolution of the standard deviation of the optimality score for a given n over ten runs. Figure 3 (b) shows the average final optimality score for a given n over ten runs. The initial parameters are the same for all runs: $|i| = 3$, $|\mathcal{A}| = 9$, $N = 80$, $K = 1$, $H = 0.4$, $\nu = 0.01$, $\mu = 10^{-4}$, and $\Phi = 0.99$. The error bars show the average one standard deviation spread of final optimality scores over the ten runs (to measure the rate of convergence).

Experiment 3: Varying the number of amino acids
The result of this experiment is that variations on $|\mathcal{A}|$ display convergence. It is important to consider that the randomly generated values for topological amino acid distance a_d and site frequency L_s will also vary, as they are generated with consideration on $|\mathcal{A}|$. The results for this are given in Figure 4. There appears to be an upwards trend in Figure 4 (b). This result is intuitively expected as increasing the number of amino acids increases the number of terms we sum over; however, further investigation is needed to confirm this. This should be done keeping randomly generated variables fixed where possible.

3.3 Varying parameters for the innovation pool structure

Recall that the algorithm from Section 2.3 begins by constructing N objects each with a genetic code G. At each time step, one of these objects receives a fragment of genome from K donors selected from the set of objects. The parameter H computes the fraction of the recipient genome due to horizontal gene transfer.

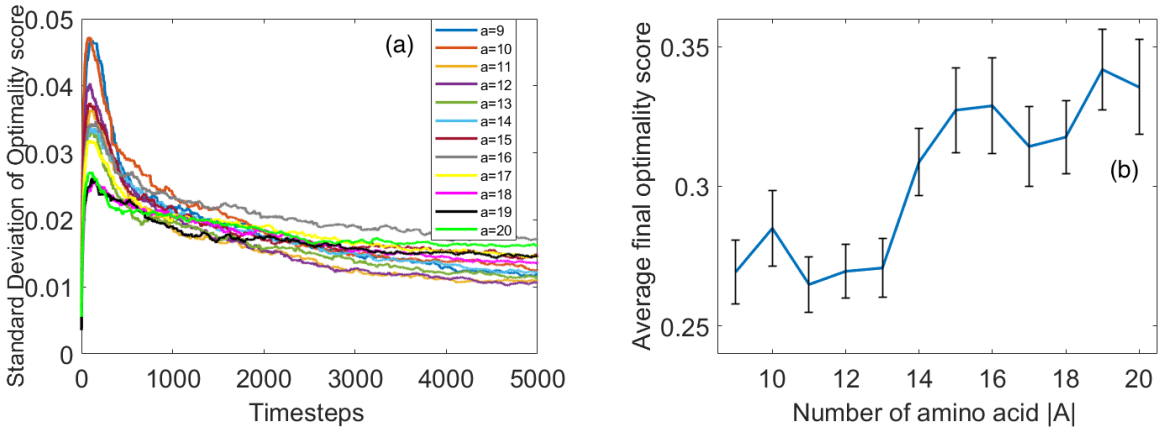


Figure 4: Figure 4 (a) shows the average time evolution of the standard deviation of the optimality score for a given $|\mathcal{A}|$ over ten runs. Figure 4 (b) shows $|\mathcal{A}|$ (number of amino acids) against the average final optimality score, averaged over ten runs. The initial parameters are the same for all runs: $|i| = 3$, $n = 3$, $N = 80$, $K = 1$, $H = 0.4$, $\nu = 0.01$, $\mu = 10^{-4}$, and $\Phi = 0.99$. The error bars in show the average one standard deviation spread of final optimality scores over the ten runs (to measure the rate of convergence).

Experiment 4: Varying N

The results for varying N is shown in Figure 9 in Appendix B. The number of entities N is taken from 10 to 100 in steps of 10. The results when varying the number of entities show a linear relationship between number of entities N and final average final optimality score as seen in Figure 9 (b).

Experiment 5: Varying K

When varying the number of donors K from 1 to 5, we find that it does not affect the algorithm's dynamics as seen in Figure 10 in Appendix B. This makes sense as we are always adding the same fraction H to the acceptor codon usage $U^A_{c,s}$. We should note there does appear to be a slight upwards trend in average final optimality score as shown in Figure 10 (b). Further investigation should be undertaken with larger value of K such that $K \approx N$.

Experiment 6: Varying H

We take H, the fraction of the genome similar due to horizontal gene transfer from 0 to 1 in increments of 0.1. The results are shown in Figure 5 below. Figure 5 (a) that for $H = 0$, there is no convergence as expected, while for $H = 0.2$ the results begin to converge but at a very slow rate. Looking at the Figure 5 (b), it is clear

that $0.4 \leq H \leq 0.7$ gives the minimal optimality score and the smallest error bars. This implies the final results have been substantially optimised and have converged to a greater extent via the attractor mechanism to a near universal solution. When $H \geq 0.8$, the final scores, O, are less optimal and have converged less.This is probably due to a change from "mixing" to "swapping" of codon usage matrices $U_{c,s}$, preventing optimal communication. Results from Figure 5 suggest that maximum mixing occurs around $H = 0.6$.

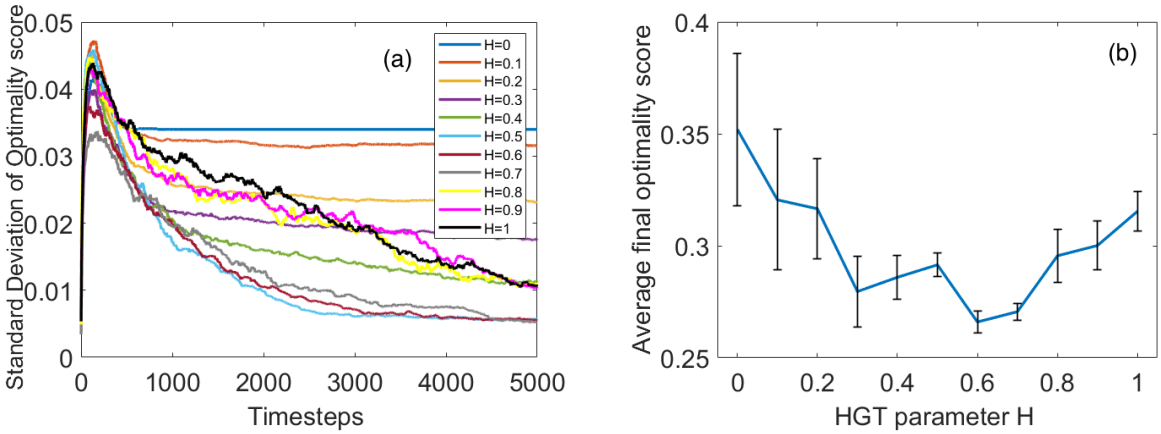


Figure 5: The average final optimality score for a given H. Figure 5 (a) shows the average time evolution of the standard deviation of the optimality score for a given H over ten runs. Figure 5 (b) shows the average final optimality score for a given H over ten runs. The initial parameters are the same for all runs: $|i| = 3$, $n = 3$, $|\mathcal{A}| = 9$, $N = 80$, $K = 1$, $\nu = 0.01$, $\mu = 10^{-4}$, and $\Phi = 0.99$. The error bars show the average one standard deviation spread of final optimality scores over ten runs (to measure the rate of convergence).

Experiment 7: Time evolution of H

The parameter H is best considered a variable that decreases with time [1, 21]. This is due to better translation of the model allowing evolution of a protein network with more specific interactions to occur [1]. To model this, we define H in the following manner:

$$H(t) = H_0 e^{-kt} \ . \tag{3.1}$$

In this equation, H_0 is the initial fraction of horizontal gene transfer similar ($0 \leq H_0 \leq 1$), and k is a constant. Initially, H_0 is set to 1. Setting $k = 10^{-3}$ gives a number that is approximately zero at, say, $t = 5000$. The results in Section 3 indicate that we expect convergence to occur after 5000 iterations. For this reason, we set $k = 10^{-4}$ so that $H(t = 5000)$ is relatively far from zero. The resulting dynamics is given in the

Figure 6 and Appendix C, where the degree of convergence is significantly improved in several runs, to all prior results. The majority of the trials have fully converged and can be considered universal. Note that the solutions converge to different values, due to a_d being randomly generated. We can see that only Figure 6 (d), does not completely converge. The rate of convergence seem to to be fairly similar to the runs when using large constant H. This makes sense as this is a stochastic process meaning the rate of convergence should vary between runs. However, the average rate of convergence significantly improves in the cases where universality manifests within this time frame. Note that $H(t)$ sits in the optimal range suggested in Section 3.3 for approximately the last 1500 time steps.

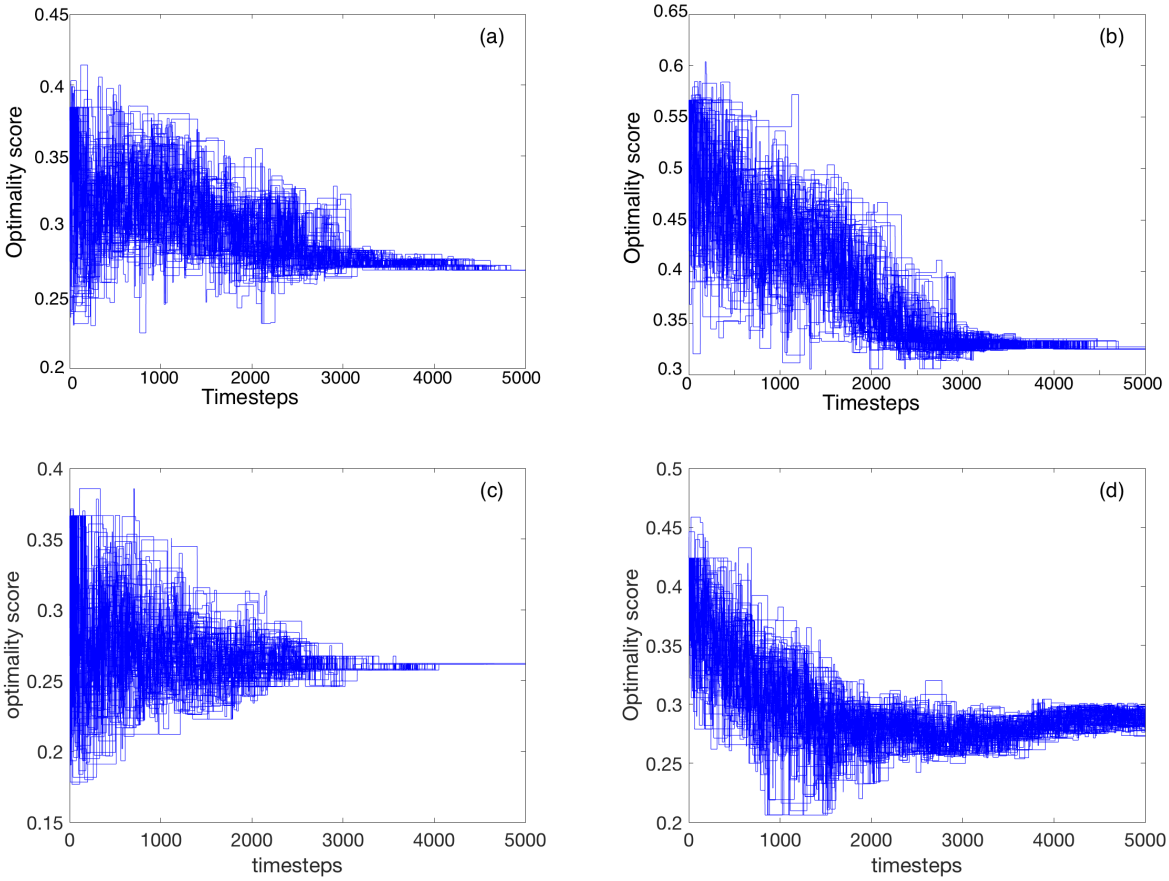


Figure 6: Evolution of optimality score with a time evolving parameter H. Four runs showing the evolution of optimality score for a time evolving H (according to (3.1)). We use: $|i| = 3$, $n = 3$, $|A| = 9$, $N = 80$, $K = 1$, $\nu = 0.01$, $\mu = 10^{-4}$, $\Phi = 0.99$, $k = 10^{-4}$, and $H_0 = 1$.

3.4 Varying error and fitness parameters

Recall that the parameters ν, μ, and Φ take values in the interval $[0, 1]$. The parameter ν measures the rate of mistranslation, when a single base is misread. The parameter μ measures the rate of mutation, when a single base is changed. The parameter Φ characterizes how an abstract physicochemical distance between amino acids scales into the fitness.

Experiment 8: Varying ν and μ
The plots for these variations are in Appendix D. It can be seen that variations on ν have no effect on a given optimality score. As we are trying to minimize the effects of errors from ν and μ, there is less requirement to optimize as they decrease. This can be seen in Figure 12 (b) in Appendix D. As these parameters decrease the optimality score increases. Note we take ν and μ from 1 to 10^{-4} on a log scale. We do not try $\nu, \mu = 0$ as this implies there is no need to optimize the code as no errors can occur.

Experiment 9: Varying Φ
As described by Vetsigian [2], Φ is a scale for the fitness for one amino acid substitution. This implies that it should not affect the rate of convergence directly. However, it will affect the score converged to. To examine this we reduce the fitness score f to a function of Φ and ν in order to consider their role in the algorithm given by Figure 13 (a) in Appendix D. The rest of the values are randomly generated. The variations of Φ are proportional to f as expected.

3.5 Re-defining universality within a genetic code model

The framework we have described so far shows that there is some degree of convergence via an attractor mechanism.With horizontal gene transfer turned on ($H \neq 0$), we have an attractor. While the details of the solution depend in part on the initial conditions assigned to parameters in the model, the model exhibits near universality at late times. This is demonstrated by the converging behaviour of the optimality scores O of all entities.We aim to refine the concept of universality. To do this we must first understand genetic code configurations and the possible symmetries associated with them. We will then analyze the fitness landscape of all genetic code configurations in order to to see if this function can be scaled homogeneously. To re-examine universality further, we mainly consider the model provided by Sella and Ardell [11] while also incorporating the fitness function provided by Vetsigian et al. [1].

Note that for this section we will also work with the fitness in the following form:

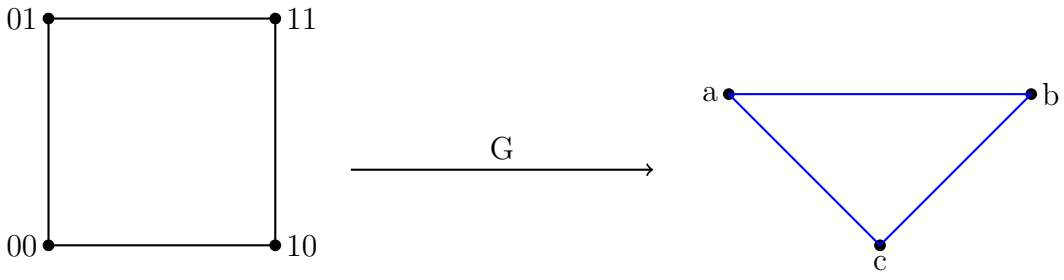
$$\log f = \sum_c \sum_s L_s U_{c,s} \log(\sum_{c'} \sum_a T_{c,c'} \Delta_{c',a} W_{a,s}) \ . \tag{3.2}$$

The logarithm simplifies algebraic manipulations.

Using Definition 2, we represent each genetic code configuration using a delta matrix $\Delta_{c,a}$. We can also calculate total number of configurations using (2.2). This framework allows us to consider the genetic code mapping as a surjective mapping from a Hamming graph (of codons) to a random graph (of distance between amino acids in an abstract topological information space). These graphs have automorphisms due to labeling which we will highlight clearly in an upcoming example. The automorphisms imply that certain genetic code configurations (and therefore delta matrices $\Delta_{c,a}$) are isomorphic to each other, meaning that they represent the same genetic code map G even though they have different delta matrices $\Delta_{c,a}$. Considering the random graph is randomly generated, we *a priori* assume that no automorphisms exist within the amino acid graph. Note this is only true for $|\mathcal{A}| > 2$, as $|\mathcal{A}| = 1$ is trivial and $|\mathcal{A}| = 2$ has an inherent symmetry in swapping the labels. Now the codons graphs as setup as a Hamming graph. Hamming graphs are known for having automorphisms [17]. Due to there being a certain number of automorphisms for the Hamming graph for a given $|i|$ and n, we quotient (2.2) by the number of symmetries to get the number of unique codes. As previously noted in Section 2.2, these automorphisms imply that two isomorphic genetic codes should yield the same optimality score.

3.5.1 Example

In order to understand the isomorphisms, we will consider an example. Put $|i| = 2$, $n = 2$, and $|\mathcal{A}| = 3$. This means $|\mathcal{C}| = 4$ giving delta matrices with dimensions 4×3. Using (2.2) we get $\#_{\text{config}}(|\mathcal{C}| = 4, |\mathcal{A}| = 3) = 36$. We represent this map in the following format:



(3.3)

In (3.3), $G : \{00, 10, 11, 01\} \mapsto \{c, c, b, a\}$. We see that the Hamming graph on the left hand side is isomorphic under relabeling [17]. In particular, if we relabel $0 \longleftrightarrow 1$,

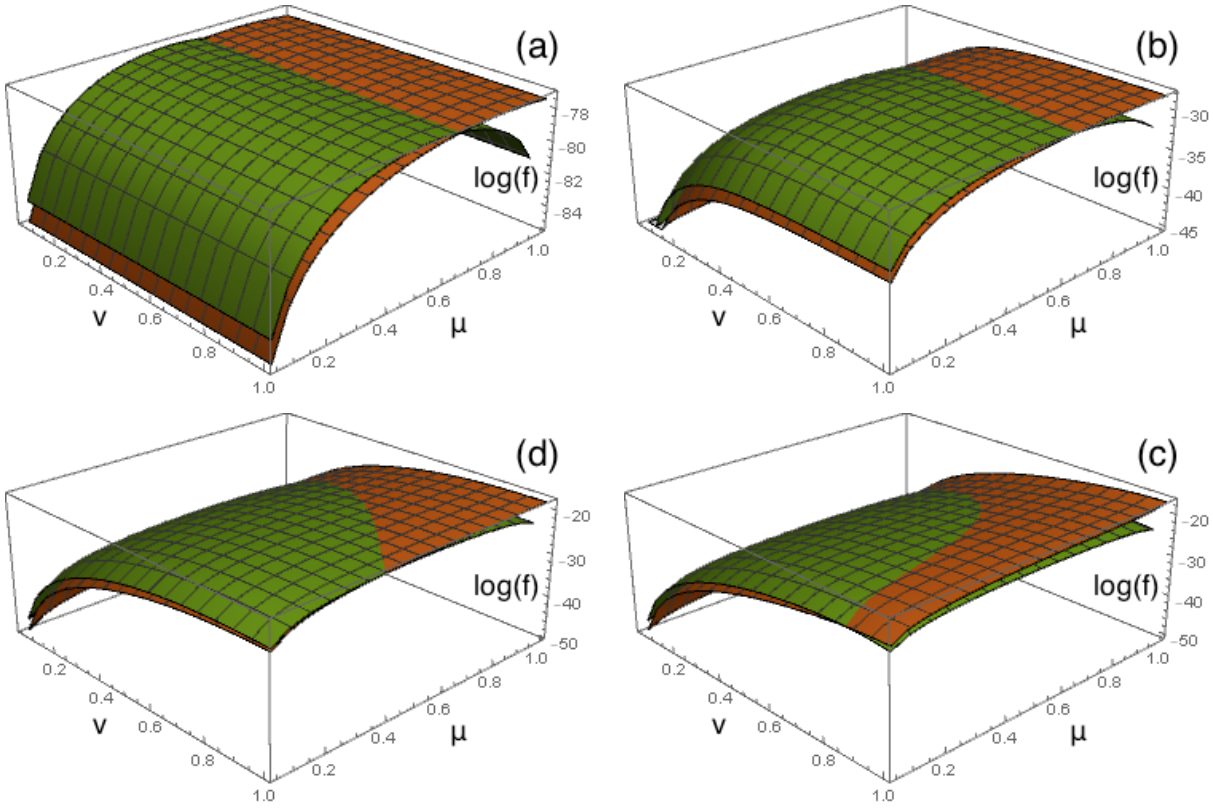


Figure 7: Four surface plots of $\log f$ in μ–ν phase space for $\Phi = 0.99$ (a), 0.5 (b), 0.1 (c), 0.01 (d). We have set $|\mathcal{C}| = 4$ and $|\mathcal{A}| = 3$. Each colored surface corresponds to a unique genetic code configuration.

the genetic code map would not change. This configuration has a symmetry factor 18. Taking the quotient of the number of configurations with the symmetry factor suggests that there are only two unique configurations of $\Delta_{c,a}$ for $|\mathcal{C}| = 4$ and $|\mathcal{A}| = 3$. Said another way, in this example, there are $\binom{4}{2}$ ways of selecting a pair of codons that are mapped by G to the same amino acid. Taking into account the repetition, there are 3! (the order of S_3) ways of mapping the codons to the amino acids. The product of these terms gives the 36 configurations. Taking into account the isomorphisms, we pick out the odd and even elements of the permutation group S_3 as our distinguished configurations.

We will now calculate (3.2) for all configurations of $\Delta_{c,a}$. We do this for a given value of Φ and generate plots in the μ–ν phase space plane (error space). In Figure 7, we show results for $\Phi = 0.99, 0.5, 0.1, 0.01$. We expect each unique genetic code configuration to correspond to a unique surface in error space.

From Figure 7, we see that there are two unique surfaces for a given value of Φ. This is due to there being two unique configurations of $\Delta_{c,a}$. Within this phase space, there is a curve on which the surfaces interact. These curves are a critical locus for which the ability of a code to produce a maximum $\log f$ changes. As Φ varies, the shape of the surfaces change and critical locus changes. In the case for $\Phi = 0.99$ (Figure 7 (a)), the critical locus is essentially independent of ν. The dependence on ν for the surfaces and the critical locus grow as Φ decreases. In order to verify this, we take a polynomial fit to the critical locus for this model.

Taking the ansatz,

$$\log(f)_{\text{fit}} = a + b_1\nu + b_2\mu + c_1\nu^2 + c_2\nu\mu + c_3\mu^2 \ , \tag{3.4}$$

for $\Phi = 0.99$ the surfaces have polynomials of the form:

$$\log(f)_{\text{fit}} = -84.8470 + 0.0801\nu - 0.0618\nu^2 + 22.0655\mu - 15.2344\mu^2 \ , \tag{3.5}$$

$$\log(f)_{\text{fit}} = -84.0745 + 0.0522\nu - 0.0404\nu^2 + 26.1504\mu - 23.5190\mu^2 \ , \tag{3.6}$$

with $R^2 > 0.99995$. Taking the difference between (3.5) and (3.6), we get the critical locus

$$-0.7725 + 0.0278\nu - 0.0214\nu^2 + 4.0849\mu + 8.2846\mu^2 = 0 \ . \tag{3.7}$$

As inferred from Figure 7, the dependence on ν is negligible as the coefficients are two or three orders of magnitude smaller than the coefficients for terms involving μ. For any value of Φ, the coefficient of the $\mu\nu$ cross term $\mathcal{O}(10^{-10})$. Thus, at $\Phi \approx 1$,

$$\frac{\partial \log f}{\partial \nu} = 0 \ . \tag{3.8}$$

This relation does not necessarily hold for smaller values of Φ for which we report results in Appendix E. Here, the coefficients of ν are on a similar magnitude to those for μ. This implies that Φ influences the effects of mistranslations ν in an inversely proportional manner. For the results in the prior sections we use $\Phi = 0.99$ for all runs as in [1]. This is due to the fact that the effects of mistranslations are more likely to be non-lethal. Note that as Φ and μ are related through eigenvectors and therefore not linearly related.

Note we also have results for $|i| = 4$, $n = 1$ and$|\mathcal{A}| = 3$ such that we have another case with $|\mathcal{C}| = 4$ and $|\mathcal{A}| = 3$ but with different automorphisms in Appendix E. For this we find a unique configuration of genetic codes and therefore no critical locus. We also find that the dependence on ν increases and Φ decreases as before.

3.6 Re-examining universality

In order to understand the universality of the model in a more formal manner, we examine it in terms of Widom scaling [22]. In particular, we look for homogeneous behavior as a signal of scale invariance on a critical locus. Consider the logarithm of the fitness function, $\log f$. As above, $f(\nu, \mu)$ is a function of the rate of mistranslations and the rate of mutations. Now, homogeneity of $\log f$ demands that

$$\log f(\kappa\nu, \kappa\mu) = \kappa^{\beta} \log f(\nu, \mu) \ , \tag{3.9}$$

where $\kappa \in \mathbb{R}$ is a scale and β is the degree of homogeneity. As $\nu, \mu \in [0, 1]$, we require that $\kappa\nu, \kappa\mu \in [0, 1]$. In particular, when $\kappa = 1$,

$$\nu \frac{\partial \log f}{\partial \nu} + \mu \frac{\partial \log f}{\partial \mu} = \beta \log f(\nu, \mu) \ . \tag{3.10}$$

This is the content of Euler's homogeneous function theorem.

However, if we consider the case of $\Phi \approx 1 - \epsilon$, where $\epsilon \ll 1$, then contributions from ν become negligible such that we can apply (3.8), leaving us to calculate $\frac{\partial \log f}{\partial \mu}$. From (3.2), the only part of the $\log f$ that depends on μ is $U_{c,s}$. Therefore, we must calculate $\frac{\partial U_{c,s}}{\partial \mu}$.

Suppose A is a real symmetric matrix with eigenvalues λ_i and eigenvectors $\mathbf{v}_i$ such that $\mathbf{v}_i^T \mathbf{v}_i = 1$. The Perron–Frobenius theorem ensures that the matrix A has a unique real eigenvalue with a magnitude larger than that of any other eigenvalue and a corresponding eigenvector with positive components. Then

$$\partial \mathbf{v}_i = (\lambda_i \mathbf{1} - A)^{+} (\partial A) \mathbf{v}_i \ , \tag{3.11}$$

where X^{+} denotes the Moore–Penrose inverse of X [23]. In defining $U_{c,s}$, we have normalized so that $\sum_c U_{c,s} = \mathbf{1}_s$. As $Q^s_{c,c'}$ is a symmetric and real matrix, we therefore only need to rescale $U_{c,s} \to U'_{c,s}$ such that $\sum_c U'_{c,s} \cdot U'_{c,s} = 1$ for any given s. By doing this we can differentiate $U'_{c,s}$:

$$\beta = \frac{\sum_c \sum_s \mu((\lambda_s^{\max} \mathbf{1} - Q^s_{c,c'})^{+} (\sum_{c''} \frac{\partial M_{c,c''}}{\partial \mu} \delta_{c'',c'} F_{c'',s}) U'_{c,s})' L_s}{\sum_c \sum_s U_{c,s} L_s} \ , \tag{3.12}$$

where

$$\frac{\partial M_{c,c''}}{\partial \mu} = \begin{cases} 1/(n(|i| - 1)) & \text{if } \mathrm{dist}(c, c') = 1 \ , \\ -1 & \text{if } \mathrm{dist}(c, c') = 0 \ , \\ 0 & \text{otherwise} \ . \end{cases} \tag{3.13}$$

We have derived the degree of scaling in the limit $\Phi \to 1$. This implies that the model is approximately homogeneous in the regime that we have worked in, with degree specified by (3.12). We see that the order is dependent on the mutations μ, implying that the universality of the code arise due to species having similar mutational errors. We regard the scaling behavior as a phenomenological observation about the solution near an approximate fixed point of the renormalization group.

We emphasize that when investigating what happens as we tweak the parameters of the model, we calculate the standard deviation to establish which results are significant when we measure the optimality score. In assessing Widom scaling, we look at the degree of homogeneity, not the degree of optimality; we do this using the fitness score, which is the function actually being maximized in this algorithm, not the optimality score. The L_s terms act as a weighting for each amino acid s. We have used a scaling argument in order to justify universality of the algorithm. The fact that the degree of homogeneity is independent of ν, the rate of mistranslations, in the $\Phi \to 1$ limit, does not imply that the final optimality score is independent of mistranslations. As seen in Experiment 8 in Section 3.4, it is not: decreasing ν increases the optimality score. We should emphasise that for smaller values of Φ, (3.12) will not hold. This is because (3.8) begins to break down for values of Φ outside the condition $\Phi \approx 1$. We can amend (3.12) to consider results around $\Phi = 0.9$ by incorporating the addition on an error term through considerations of the first term of (3.10). This approximation does not apply for smaller values of Φ, however, since, as suggested by Sella and Ardell [11], we expect Φ to be relatively large.

4 Conclusion and prospects

In this paper, we have argued that with generic initial conditions, there is a late time near universality resulting from the flow of the theory to an attractive solution, *viz.*, the standard genetic code. The convergence via the attractor mechanism to a near universal solution relies on the mechanism of horizontal gene transfer [1], which corresponds to setting a parameter H to a non-zero value. We varied the parameters of the model and found that all variations still display this convergence, except for $H = 0$. This demonstrates the robustness of the model. We also found that for $0.3 < H < 0.7$ we obtain near universal solutions with the greatest degree of optimisation, with $H > 0.7$ not being as effective due to some transition from "mixing" to "swapping". Taking H as a decreasing time-dependent function vastly improves the convergence in comparison to constant values of H. We found that increasing the number of codons, $|\mathcal{C}|$, increases

the optimality score O. By limiting the fitness function to a regime that we work in ($\Phi = 0.99$), we are able to make approximations that lead to homogeneity in $\log f$, where $f(\nu, \mu)$ is a fitness function depending on the rate of mistranslation and the rate of point mutation. We derive an expression for the degree of homogeneity, β. In the limit $\Phi \to 1$, β depends strongly on the mutation rate and negligibly on the mistranslation rate. We conclude that the rate of point mutations is the crucial factor in driving arbitrary initial conditions to the attractor solution that optimizes fitness of the genetic code. The point mutation rate is determined by the eigenvalue of the linearized renormalization group transformation around the fixed point for the dynamics.

Improvements to make the algorithm more accurate for biology would involve exploring how to incorporate stop codons which do not code for amino acids into the model as something more than a dummy amino acid. We should also consider that mutations and horizontal gene transfer do not occur at the same rate as suggested by both occurring at each iteration. Some work to estimate a timescale for this model possibly by considering rate of error as the sum of all errors ($\nu + \mu$) and relating this to the measured rate of error in, for example, a kinetic proofreading model [24]. We also suggest that additional factors and steps should be incorporated in order to guarantee a universal solution is converged to every run.

Variations on the dimensions, $|\mathcal{C}|$ and $|\mathcal{A}|$, which count the number of codons and the number of amino acids, respectively, display convergence to a universal result. This could have applications to synthetic biology where codes with up to 8 bases have been created [25]. These codes should also converge to a universal genetic code given enough time. An open question is to determine what sets the initial conditions. Why did life on Earth evolve to make use of four base pairs in DNA, three base pairs per codon, and 20 amino acids?

We have focused on a single basin of attraction, whereas there could be others. The basin of attraction may be determined by biochemistry inputs. We can imagine, for example, a different basin of attraction in which the solvent is ammonia, methane, or hydrogen fluoride instead of water. We can also imagine biochemistry organized around silicon instead of carbon. The molecular realization of the genetic code would be different based on these other inputs, but we expect that the same principles apply, and these other hypothetical genetic codes would also evolve to a universal solution based on the principle of horizontal gene transfer.

Broadly speaking, we have argued that the concept of universality from statistical physics applies to biological systems like the genetic code. The thermodynamic limit arises from a large N number of degrees of freedom in the entities studied. The dy-

namical system is driven to an attractor solution as a result of interactions, in this case horizontal gene transfer. We have considered a mechanism for horizontal gene transfer and by tweaking its parameters identified which ones are the most important. The existence of approximate homogeneity offers evidence for universality. We would like to interrogate how general this setup is and whether it is useful for studying other complex systems.

Indeed, like thermodynamics and evolution itself, we wish to consider horizontal gene transfer as an organizing principle in Nature. Different solutions to a theory or different possible initial conditions can exchange information with each other through complex processes. Dynamics can then flow the system to a late time attractor solution that is independent of specific parameters of the model. As a proving ground for this hypothesis, we can consider the vacuum selection problem in quantum gravity. String theory, a promising candidate framework for marrying gravitation with quantum theory, generically predicts a landscape of vacua, one of which is our Universe with the Standard Models of particle physics and cosmology as phenomenological features that explain dynamics at small and large scales. (See, for example, [26–28] for related reviews.) These vacua inevitably arise as a consequence of a simple observation — we live in four spacetime dimensions whereas the consistency of the theory demands ten, and there is no unique way to reduce the number of dimensions. Because they are unobserved, the extra dimensions predicted by string theory comprise a compact geometry with special properties. The moduli space of string compactifications is believed to be connected. We can calculate the degree of fine tuning necessary to support certain cosmological structures and the astrophysical and chemical preconditions necessary for life [29]. Rather than making an explicitly anthropic argument [30, 31], we can model a dynamics for vacuum selection which incorporates a mechanism analogous to horizontal gene transfer to lead to universal and optimal structures as an attractive fixed point. Thus, instead of arguing that low energy observables such as the cosmological constant are distributed randomly across the landscape, horizontal gene transfer, by driving the system to the attractor value, may obviate aspects of the measure problem. Developing and testing this hypothesis within the string theory framework is work in progress.

Acknowledgements

We are grateful to David Ardell, Nigel Goldenfeld, Sujay Nair, and Kalin Vetsigian for feedback and insightful discussions. YHH is indebted to the Science and Technology Facilities Council, UK, for grant ST/J00037X/1. VJ thanks the South African Re-

search Chairs Initiative of the Department of Science and Technology and the National Research Foundation for support. DM thanks the Julian Schwinger Foundation and the US Department of Energy (DE-SC0020262) for support.

A Different initial delta matrices $\Delta_{c,a}$

Initializing with different delta matrices, the optimality score converges.

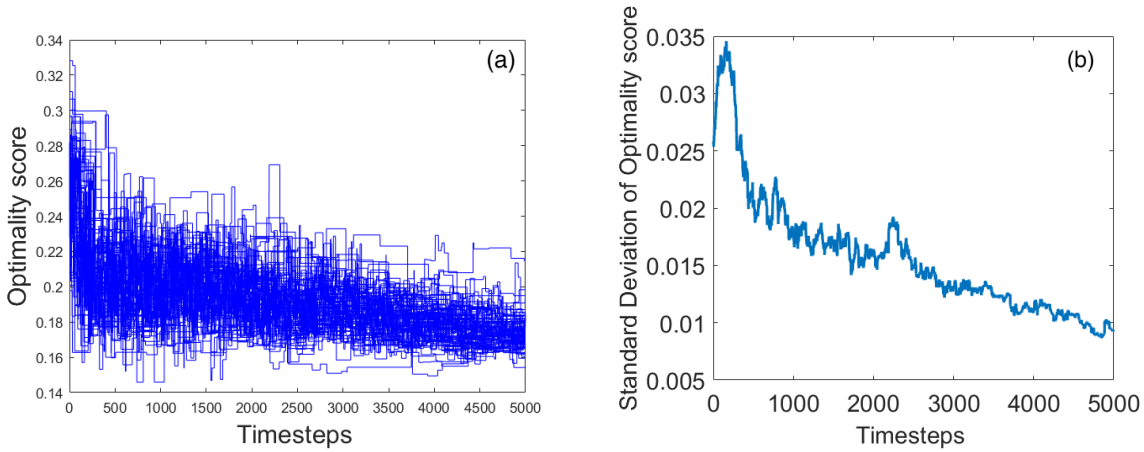


Figure 8: Graph showing evolution of optimality score when all entities have a different initial $\Delta_{c,a}$ rather than the same initial $\Delta_{c,a}$. Initial parameters are: $|i| = 3$, $n = 3$, $|A| = 9$, $N = 80$, $K = 1$, $H = 0.4$, $\nu = 0.01$, $\mu = 10^{-4}$, and $\Phi = 0.99$

B Varying innovation pool structure

As discussed in Section 3.3, we plot what happens as we vary N, the number of entities under consideration, and K, the number of donors.

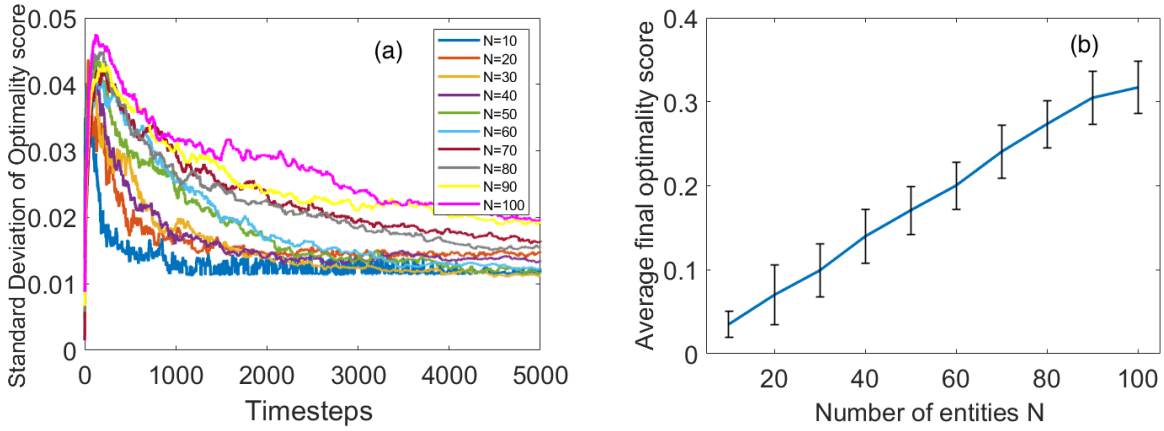


Figure 9: Figure 9 (a) shows the average time evolution of the standard deviation of the optimality score for a given N over ten runs. Figure 9 (b) shows N (number of entities) against the average final optimality score, averaged over ten runs. The initial parameters are the same for all runs: $|i| = 3$, $n = 3$, $|\mathcal{A}| = 9$, $K = 1$, $H = 0.4$, $\nu = 0.01$, $\mu = 10^{-4}$, and $\Phi = 0.99$. The error bars show the average one standard deviation spread of final optimality scores over ten runs (to measure the rate of convergence).

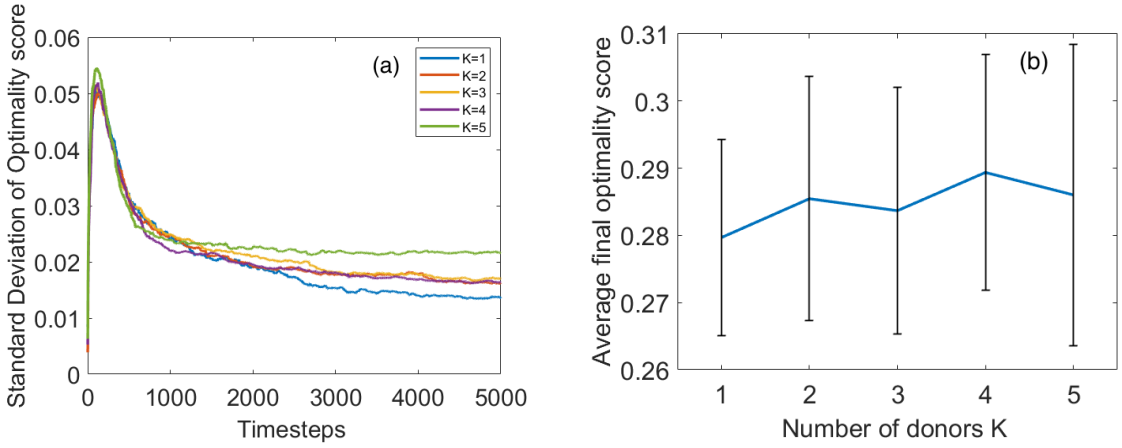


Figure 10: Figure 10 (a) shows the average time evolution of the standard deviation of the optimality score for a given K over ten runs. Figure 10 (b) shows K (number of donors) against the average final optimality score, averaged over ten runs. The initial parameters are the same for all runs: $|i| = 3$, $n = 3$, $|\mathcal{A}| = 9$, $N = 80$, $H = 0.4$, $\nu = 0.01$, $\mu = 10^{-4}$, and $\Phi = 0.99$. The error bars show the average one standard deviation spread of final optimality scores over ten runs (to measure the rate of convergence).

C Time evolution of H

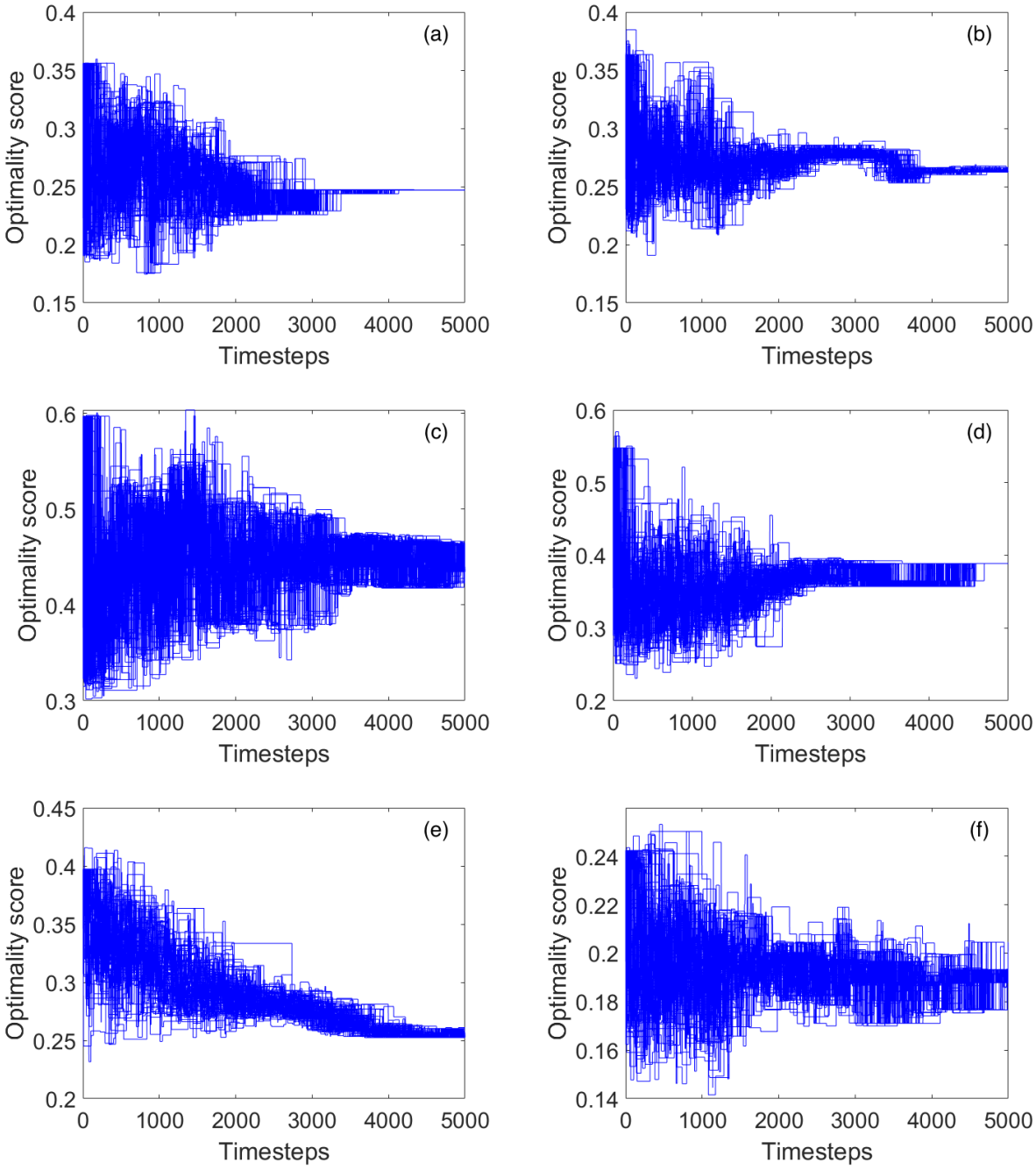


Figure 11: Evolution of optimality score with a time evolving parameter H. Six runs showing the evolution of optimality score for a time evolving H (according to (3.1)). We use: $|i| = 3$, $n = 3$, $|A| = 9$, $N = 80$, $K = 1$, $\nu = 0.01$, $\mu = 10^{-4}$, $\Phi = 0.99$, $k = 10^{-4}$, and $H_0 = 1$.

D Varying noise and fitness parameters

As discussed in Section 3.4, we plot variations of ν (the mistranslation rate), μ (the mutation rate), and Φ (the scale for abstract physicochemical distance).

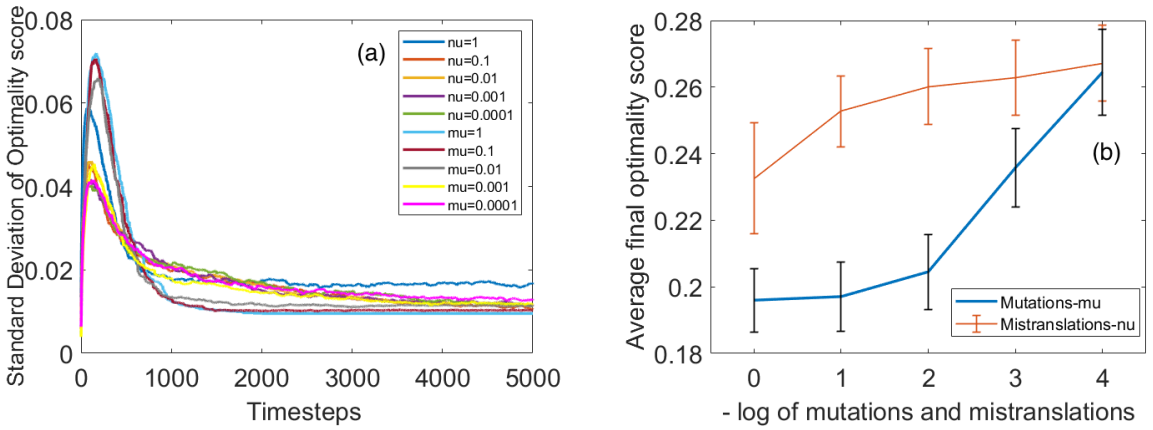


Figure 12: Figure 12 (a) shows the average time evolution of the standard deviation of the optimality score for a given ν and μ over ten runs. Figure 12 (b) shows ν and μ against the average final optimality score, averaged over ten runs. The initial parameters are the same for all runs: $|i| = 3$, $n = 3$, $|\mathcal{A}| = 9$, $N = 80$, $K = 1$, $H = 0.4$, and $\Phi = 0.99$. When varying ν, we put $\mu = 10^{-4}$. When varying μ, we put $\nu = 0.01$. The error bars show the average one standard deviation spread of final optimality scores over three runs (to measure the rate of convergence).

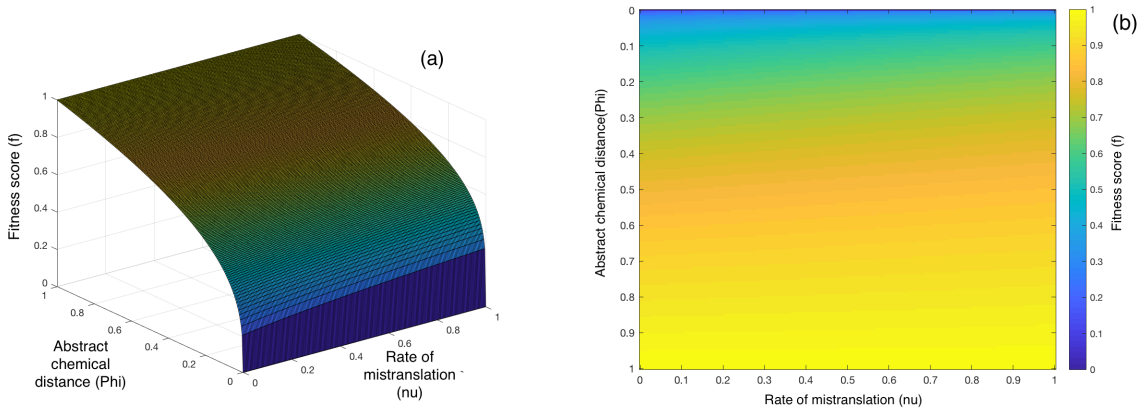


Figure 13: Figure 13 (a) displays a surface plot of the fitness function f in terms of Φ and ν. Figure 13 (b) is simply a heatmap of the first plot. We use: $|i| = 3$, $n = 3$, $|\mathcal{A}| = 9$, $N = 80$, $K = 1$, $H = 0.4$, and $\mu = 10^{-4}$. Other parameters are randomly generated.

E Defining universality

We show the best fits for different values of Φ corresponding to the surfaces for $\log f$ as a function of ν and μ in Figure 7.

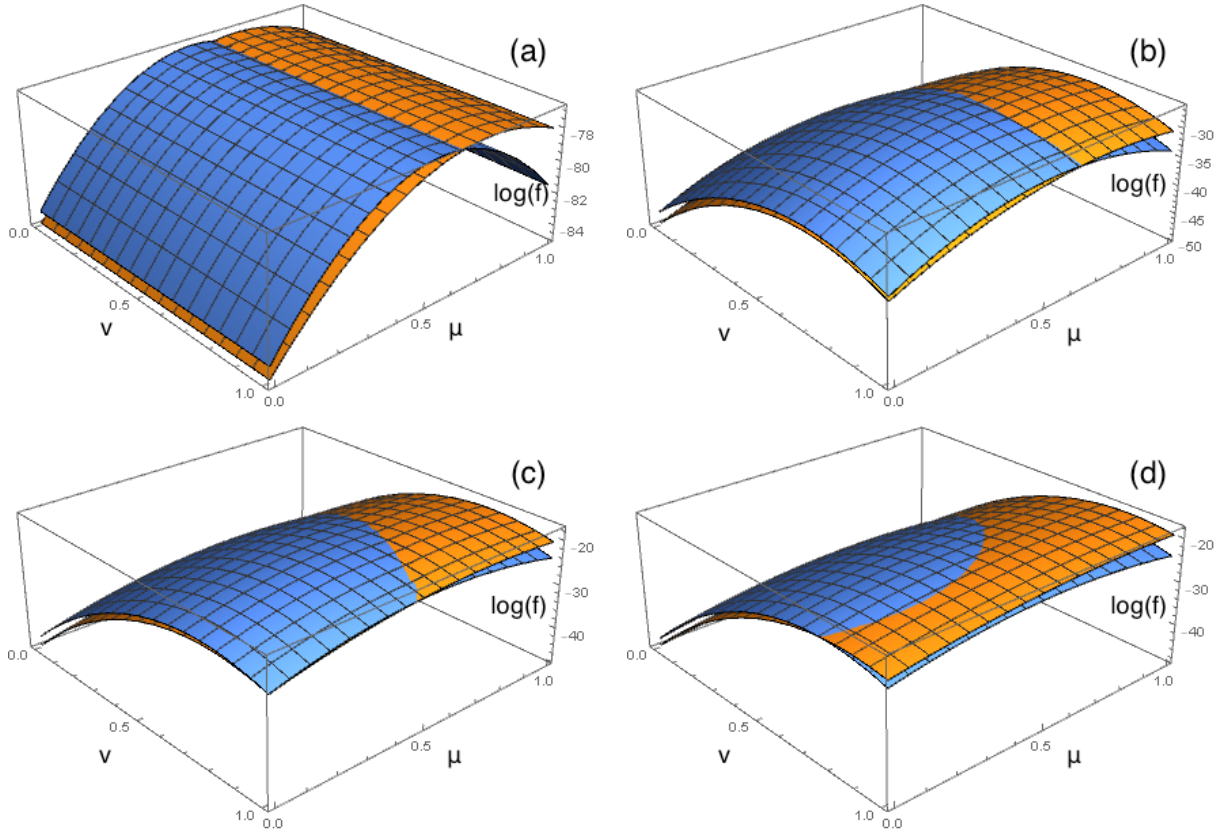


Figure 14: Four surface plots of polynomial fit of $\log f$ in μ–ν phase space for $\Phi = 0.99$ (a), 0.5 (b), 0.1 (c), 0.01 (d). This is to be compared with Figure 7.

In analogy to (3.5) and (3.6) for $\Phi = 0., 99$, we provide polynomial best fits for $|i| = 2$, $n = 2$ and $|\mathcal{A}| = 3$ for $\Phi = 0.5, 0.1, 0.01$.

- $\Phi = 0.5$**:**

$$\log(f)_{\text{fit}} = -49.6633 + 44.1899\nu - 31.6042\nu^2 + 21.4761\mu - 14.8319\mu^2\,, \qquad R^2 = 0.99761\,. \tag{E.1}$$

$$\log(f)_{\text{fit}} = -47.5384+40.8086\nu-29.5881\nu^2+25.3872\mu-22.8529\mu^2\,, \qquad R^2 = 0.997706\,. \tag{E.2}$$

- $\Phi = 0.1$**:**

$$\log(f)_{\text{fit}} = -47.7527+61.5566\nu-39.4707\nu^2+16.8343\mu-11.4213\mu^2\,, \qquad R^2 = 0.99705\,, \tag{E.3}$$

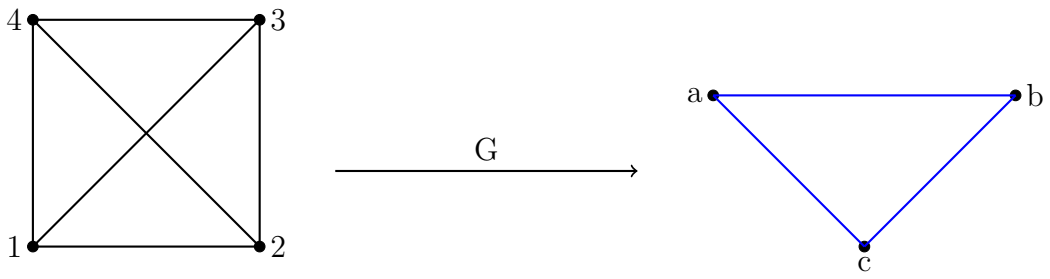
$$\log(f)_{\text{fit}} = -45.5857+59.7551\nu-39.8311\nu^2+19.962\mu-18.1142\mu^2\,, \qquad R^2 = 0.996994\,. \tag{E.4}$$

- $\Phi = 0.01$**:**

$$\log(f)_{\text{fit}} = -48.6657+62.1467\nu-36.9472\nu^2+11.0255\mu-6.69862\mu^2\,, \qquad R^2 = 0.997404\,, \tag{E.5}$$

$$\log(f)_{\text{fit}} = -46.9935+63.104\nu-41.1589\nu^2+14.3666\mu-13.2086\mu^2\,, \qquad R^2 = 0.997725\,. \tag{E.6}$$

The values $|i| = 4$, $n = 1$, $|\mathcal{C}| = 4$, $|\mathcal{A}| = 3$ is represented by the map:



(E.7)

These give the surface plots of polynomial fit of $\log f$ over the μ–ν phase space shown in Figure 15.

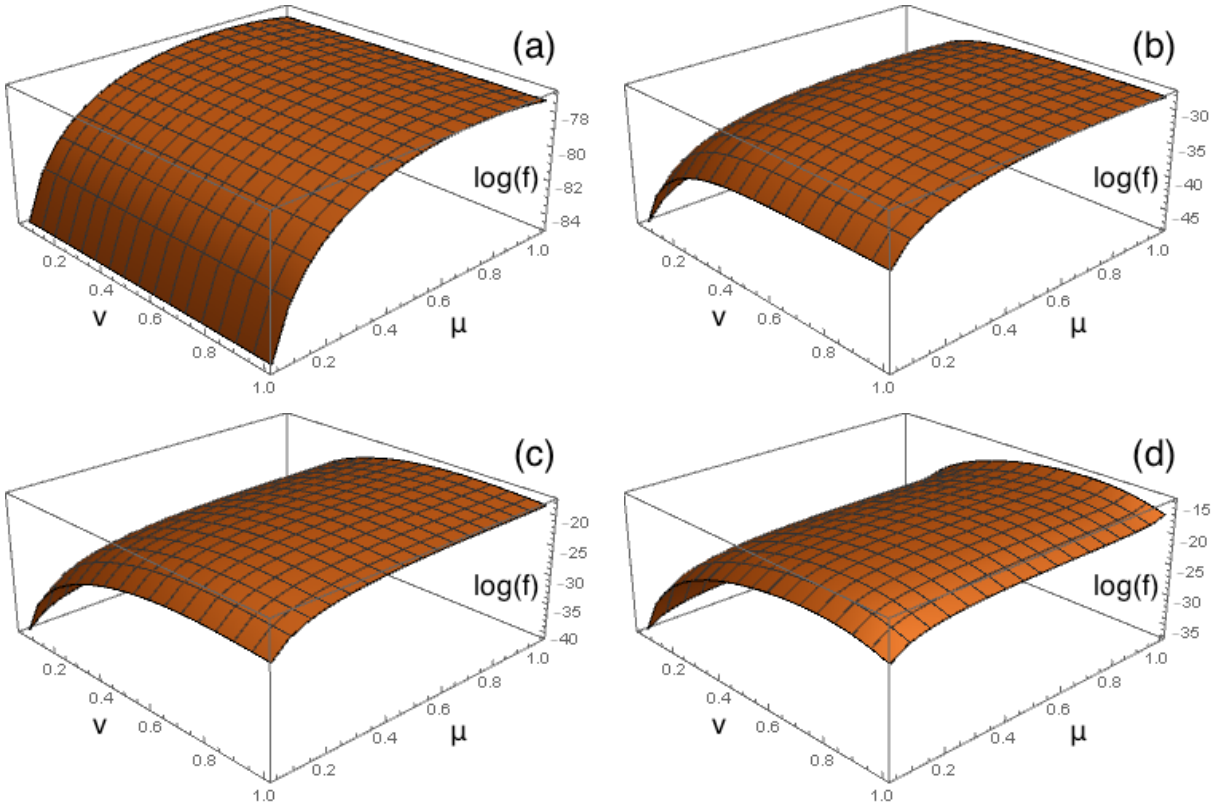


Figure 15: Four surface plots of polynomial fit of log f in μ–ν phase space for $\Phi = 0.99$ (a), 0.5 (b), 0.1 (c), 0.01 (d).

References

[1] Kalin Vetsigian, Carl Woese, and Nigel Goldenfeld. Collective evolution and the genetic code. *Proceedings of the National Academy of Sciences*, 103(28):10696–10701, 2006. ISSN 0027-8424. doi: 10.1073/pnas.0603780103. URL https://www.pnas.org/content/103/28/10696.

[2] Kalin Horen Vetsigian. Collective evolution of biological and physical systems, 2005. URL http://guava.physics.uiuc.edu/people/Theses/Vetsigian_PhD_thesis.pdf.

[3] Guy Sella and David H. Ardell. The coevolution of genes and genetic codes: Crick's frozen accident revisited. *Journal of Molecular Evolution*, 63(3):297–313, Sep 2006. ISSN 1432-1432. doi: 10.1007/s00239-004-0176-7. URL https://doi.org/10.1007/s00239-004-0176-7.

[4] S.G. Bonitz, Roberta Berlani, Gloria Coruzzi, May Li, Giuseppe Macino, F.G. Nobrega, M.P. Nobrega, B.E. Thalenfeld, and Alexander Tzagoloff. Codon recognition rules in yeast mitochondria. *Proceedings of the National Academy of Sciences of the United States of America*, 77:3167–70, 07 1980. doi: 10.1073/pnas.77.6.3167. URL https://www.pnas.org/content/77/6/3167.

[5] Stephen J. Freeland and Laurence D. Hurst. The genetic code is one in a million. *Journal of molecular evolution*, 47:238–48, 10 1998. doi: 10.1007/PL00006381. URL https://doi.org/10.1007/PL00006381.

[6] Stephen J. Freeland, Tao Wu, and Nick Keulmann. The case for an error minimizing standard genetic code. *Origins of life and evolution of the biosphere*, 33(4):457–477, Oct 2003. ISSN 1573-0875. doi: 10.1023/A:1025771327614. URL https://doi.org/10.1023/A:1025771327614.

[7] David Haig and Laurence D. Hurst. A quantitative measure of error minimization in the genetic code. *Journal of Molecular Evolution*, 33(5):412–417, Nov 1991. ISSN 1432-1432. doi: 10.1007/BF02103132. URL https://doi.org/10.1007/BF02103132.

[8] C.R. Woese, D.H. Dugre, Carl Saxinger, and S.A. Dugre. The molecular basis for the genetic code. *Proceedings of the National Academy of Sciences of the United States of America*, 55:966–74, 05 1966. doi: 10.1073/pnas.55.4.966. URL https://www.pnas.org/content/55/4/966.

[9] Damien C. Mathew and Zaida Luthey-Schulten. On the physical basis of the amino acid polar requirement. *Journal of Molecular Evolution*, 66(5):519–528, May 2008. ISSN 1432-1432. doi: 10.1007/s00239-008-9073-9. URL https://doi.org/10.1007/s00239-008-9073-9.

[10] Thomas Butler, Nigel Goldenfeld, Damien Mathew, and Zaida Luthey Schulten. Extreme genetic code optimality from a molecular dynamics calculation of amino acid polar requirement. *Physical review. E, Statistical, nonlinear, and soft matter physics*, 79:060901, 07 2009. doi: 10.1103/PhysRevE.79.060901. URL https://link.aps.org/doi/10.1103/PhysRevE.79.060901.

[11] Guy Sella and David H. Ardell. The impact of message mutation on the fitness of a genetic code. *Journal of Molecular Evolution*, 54(5):638–651, May 2002. ISSN 1432-1432. doi: 10.1007/s00239-001-0060-7. URL https://doi.org/10.1007/s00239-001-0060-7.

[12] S.H. Strogatz. *Nonlinear Dynamics and Chaos: With Applications to Physics, Biology, Chemistry and Engineering*. Studies in nonlinearity. Westview, 2000. URL https://books.google.co.uk/books?id=NZZDnQEACAAJ.

[13] Kenneth G. Wilson. The Renormalization Group: Critical Phenomena and the Kondo Problem. *Rev. Mod. Phys.*, 47:773, 1975. doi: 10.1103/RevModPhys.47.773. URL https://link.aps.org/doi/10.1103/RevModPhys.47.773.

[14] K. G. Wilson. The renormalization group and critical phenomena. *Rev. Mod. Phys.*, 55:583–600, 1983. doi: 10.1103/RevModPhys.55.583. URL https://link.aps.org/doi/10.1103/RevModPhys.55.583.

[15] N. Goldenfeld. *Lectures on phase transitions and the renormalization group*. 1992. ISBN 9780429973123. URL https://books.google.co.uk/books?id=HQpQDwAAQBAJ.

[16] D.J. Baylis. *Error Correcting Codes: A Mathematical Introduction*. Taylor & Francis, 2017. ISBN 9781351449847. URL https://books.google.co.uk/books?id=r0OVtAEACAAJ.

[17] S. Morteza Mirafzal and Meysam Ziaee. A note on the automorphism group of the Hamming graph. *arXiv e-prints*, art. arXiv:1901.07784, Jan 2019. URL https://arxiv.org/abs/1901.07784.

[18] W. Bialek. *Biophysics: Searching for Principles*. Princeton University Press, 2012. ISBN 9780691138916. URL https://books.google.co.uk/books?id=5In_FKA2rmUC.

[19] R. B.J.T. Allenby and Alan Slomson. *How to Count: An Introduction to Combinatorics, Second Edition*. Chapman & Hall/CRC, 2nd edition, 2010. ISBN 1420082604, 9781420082609. URL https://books.google.co.uk/books?id=iRrSBQAAQBAJ.

[20] Peter Becich, Brian P Stark, Harish S Bhat, and David Ardell. Cmcpy: Genetic code-message coevolution models in python. *Evolutionary bioinformatics online*, 9:

111–25, 02 2013. doi: 10.4137/EBO.S11169. URL https://doi.org/10.4137/EBO.S11169.

[21] Carl R. Woese. On the evolution of cells. *Proceedings of the National Academy of Sciences*, 99(13):8742–8747, 2002. ISSN 0027-8424. doi: 10.1073/pnas.132266999. URL https://www.pnas.org/content/99/13/8742.

[22] B. Widom. Equation of State in the Neighborhood of the Critical Point. *Journal of Chemical Physics*, 43:3898–3905, December 1965. doi: 10.1063/1.1696618.

[23] K. B. Petersen and M. S. Pedersen. The matrix cookbook, October 2008. URL http://www2.imm.dtu.dk/pubdb/p.php?3274. Version 20081110.

[24] U. Alon. *An Introduction to Systems Biology*. Chapman & Hall/Crc Mathematical and Computational Biology. CRC Press LLC, 2019. ISBN 9781439837177. URL https://books.google.co.uk/books?id=MWXdQgAACAAJ.

[25] Shuichi Hoshika, Nicole A. Leal, Myong-Jung Kim, Myong-Sang Kim, Nilesh B. Karalkar, Hyo-Joong Kim, Alison M. Bates, Norman E. Watkins, Holly A. SantaLucia, Adam J. Meyer, Saurja DasGupta, Joseph A. Piccirilli, Andrew D. Ellington, John SantaLucia, Millie M. Georgiadis, and Steven A. Benner. Hachimoji dna and rna: A genetic system with eight building blocks. *Science*, 363(6429):884–887, 2019. ISSN 0036-8075. doi: 10.1126/science.aat0971. URL https://science.sciencemag.org/content/363/6429/884.

[26] Frederik Denef and Michael R. Douglas. Computational complexity of the landscape. I. *Annals Phys.*, 322:1096–1142, 2007. doi: 10.1016/j.aop.2006.07.013. URL https://ui.adsabs.harvard.edu/abs/2007AnPhy.322.1096D.

[27] Vishnu Jejjala, Michael Kavic, and Djordje Minic. Time and M-theory. *Int. J. Mod. Phys.*, A22:3317–3405, 2007. doi: 10.1142/S0217751X07036981. URL https://doi.org/10.1142/S0217751X07036981.

[28] Yang-Hui He. The Calabi-Yau Landscape: from Geometry, to Physics, to Machine-Learning. 2018. URL https://arxiv.org/abs/1812.02893.

[29] Fred C. Adams. The Degree of Fine-Tuning in our Universe – and Others. 2019. URL https://arxiv.org/abs/1902.03928#.

[30] Steven Weinberg. The Cosmological Constant Problem. *Rev. Mod. Phys.*, 61:1–23, 1989. doi: 10.1103/RevModPhys.61.1. URL https://link.aps.org/doi/10.1103/RevModPhys.61.1.

[31] Joseph Polchinski. The Cosmological Constant and the String Landscape. In *The Quantum Structure of Space and Time: Proceedings of the 23rd Solvay Conference on*

Physics. Brussels, Belgium. 1 - 3 December 2005, pages 216–236, 2006. URL https://arxiv.org/abs/hep-th/0603249.