SCIENTIFIC REPORTS

OPEN

SUBJECT AREAS:
QUANTUM PHYSICS
THEORY
THEORETICAL PHYSICS
MODELLING AND THEORY

Received
18 January 2012

Accepted
16 February 2012

Published
12 March 2012

Correspondence and requests for materials should be addressed to M.A.M. (mardel@miranda.fis.ucm.es)

On Quantum Effects in a Theory of Biological Evolution

M. A. Martin-Delgado

Departamento de Física Teórica I, Universidad Complutense, 28040 Madrid, Spain.

We construct a descriptive toy model that considers quantum effects on biological evolution starting from Chaitin's classical framework. There are smart evolution scenarios in which a quantum world is as favorable as classical worlds for evolution to take place. However, in more natural scenarios, the rate of evolution depends on the degree of entanglement present in quantum organisms with respect to classical organisms. If the entanglement is maximal, classical evolution turns out to be more favorable.

Ever since its development by Darwin[1], the theory of evolution stands up as the landmark of fundamental knowledge in life sciences. In this sense, it is a theory of everything that unifies all species with a common origin. The driving principle of evolution is the 'survival of the fittest'. This leads to a common origin to all species and biological diversity. Before it, biology was conceived as static through history. After it, biological effects are given a dynamical framework.

As it stands, evolution is now considered the basic principle of biology and has the same character as a physical law: it is true as long of all pieces of experimental evidence support it. However, this does not preclude raising the fundamental question as to why living organism evolve. This question also arises in physical laws and the underlying issue is the search for more fundamental principles.

In a recent work, G. Chaitin[2–4] has challenged the status of evolution and asked the question: is it possible to give a mathematical proof of evolution? As well as why is it that living organisms evolve.

It is apparent that in addressing such deep questions one cannot take into account all the details that are present in a living organism, whether it is highly evolved or not. One needs to abstract the basic features and come up with a toy model in order to be able to work with it. Chaitin has followed this method and he uses a very basic definition of what a living organism is and a remarkable notion of a mutation. His model and insight are inspired by his earlier works on Algorithmic Information Theory (AIT)[5,6]. We will refer to it as the Chaitin model and we shall describe it in Sect. I A.

A natural and challenging problem is how to introduce quantum effects in the classical model of Chaitin, and then try to evaluate its consequences. This is the purpose of this paper. Related to this, an interesting questions is: **What is more favorable, to evolve in a quantum world or in a classical world?**

The answer to this question is relevant in several ways since it could shed some light to other fundamental questions:

i/ Biological evolution was formulated as a basic feature of classical living organisms for our world is classical at the macroscopic level. However, there could have been an earlier time previous to our current 'classical era' in which quantum effects may have played a role in evolution. Thus, was there a quantum evolution epoch before classical evolution took place?

ii/ Alternatively, there is also the possibility that classical and quantum evolution coexists at different scales. Is this possible or favorable?

A basic assumption of our quantum model for biological evolution will be the Turing barrier: a quantum computer can not compute a problem that is uncomputable for a classical computer, i.e. for a Turing machine (TM). For example, the Turing halting problem[7] is also uncomputable for a quantum Turing machine. In his famous paper on quantum simulators, Feynman's argues that this barrier is unsurmountable[8] and this is the widely accepted status on these quantum limits[9], despite several attempts to beat the Turing barrier[10,11]. We leave for the Discussion section the interesting analysis on the possible consequences of beating the Turing barrier for the quantum Chaitin model of biological evolution.

It is very deep and insightful the use of non-computability as something positive as opposed to how it is appreciated in more pragmatical approaches to the foundations of the theory of computation. In mathematics,

there is also intrinsic randomness, and Chaitin uses non-computability as a resource to have an appropriate fitness function to challenge organism to evolve, thereby improving and becoming more advanced. This is elaborated further in the Discussion section, Sec.II.

Schrödinger was the precursor of studying quantum effects in DNA[12] and he thought about the possibility that mutations were originated by some sort of quantum fluctuations. The notion of mutation introduced here, Sec.I C, is far more general.

When addressing the issue of quantum effects in Chaitin biological evolution, it is crucial to bear in mind the following fact:

i/ Complexity classes are affected by quantum effects and they are different than in the classical case.
ii/ Computability remains the same for both quantum and classical cases (this is the Turing barrier).

Thus, as the Chaitin model is based on *non-computability* as a resource for driving evolution, then apparently there should not be any quantum effects. However, the key point is that Chaitin defines an organism as a finite-size program software. Once its size N is fixed, thus being finite, it is also computable, thereby becoming a complexity problem. Thus, the way out to this apparent paradox is to realize that for finite N-size problem, there is no computability issue. What it is true is that $\forall N$, it is not possible to compute the fitness functions of Chaitin based on non-computability.

The version of algorithmic complexity introduced by Kolmogorov is not prefix-free (self-delimiting programs) and does not allow to formulate halting probabilities as in Chaitin's version of algorithmic complexity. This is why we use the latter.

This paper is organized as follows: in Sect.I A we review the classical model introduced by Chaitin to study classical evolution scenarios using the formalism and results of AIT; in Sect.I C the quantum versions of organisms, mutations and fitness functions are formulated on very general grounds; in Sect.I D a choice of quantum algorithmic complexity has to be made and we review the known results for entangled and separable quantum states; in Sect.I E we introduce quantum Ω numbers which play a central role in defining mutations in a quantum world; in Sect.I F we analyze the total evolution time and its scaling with the number of time-steps, for several quantum evolution scenarios and quantum organisms; Sect.II is devoted to discussion, prospects and further explanations. The Methods section III explains some basic notions of AIT and in particular, prefix-free bit-strings and its coding that are necessary to compute the complexity of quantum mutations.

Results

A. Classical chaitin model. *1. The model.* The fundamental notion in Chaitin model is to consider life as evolving software. This will be specified below. To this end, let us recall some basic notions from AIT that are needed to define the model. Let $\mathfrak{X}:=\{\Lambda,0,1,00,01,10,11,000,\ldots\}$ be the set of finite strings of binary bits, with Λ denoting the blank space symbol. The size or number of bits is $|x|$. The set of infinite bit-strings is denoted as $\mathfrak{X}^{\infty}$. A classical computer is an application $C:\mathfrak{X}\times\mathfrak{X}\to\mathfrak{X}$ that takes an input data $q\in\mathfrak{X}$ and a program $p\in\mathfrak{X}$ and acts on the input to produce an output string $C(p,q)=x\in\mathfrak{X}$ which is the result of the computation, assuming it halts. The concrete structure and functioning of C is given by the classical Turing Machine[5,6]. When the input data is empty, we simply write $C(p)=x$ and when the output is simply stopping the computer with no output, we write $C(p)$: halts. A universal Turing Machine (UTM) U is one that can simulate the functioning of any other TM C.

The notion of complexity is basic in computability theory. It tells us whether a program $p\in\mathfrak{X}$ or input/output data q, $x\in\mathfrak{X}$ have a simple structure or not. Throughout this paper, we shall be using the notion of algorithmic complexity $H(x)$ of a generic string of bits $x\in\mathfrak{X}$. It was studied independently by Solomonoff[13], Kolmogorov[14] and Chaitin[15], and sometimes is referred to as Kolmogorov complexity. It is defined as the shortest program that can reproduce a given string x in a universal TM:

$$H(x):=\min_{p:\,U(p)=x}|p|. \tag{1}$$

This notion of complexity grasp the concept that the information content of a string is more related to its intrinsic computational structure rather than to its mere size. For example, a string like $x=0101010101010101\ldots$ may be very large, but its structure is very simple; $x=(01)^n$, for a certain integer n. The same goes for other periodic strings or structured strings. Its complexity is bounded by a constant; $H(x)<c$. On the other side of the complexity are the random strings x_r that are those without internal structure. This is represented by a complexity $H(x_r)\geq|x_r|$, for the best thing a TM can do is to output the same input string x_r.

A remark is in order. The algorithmic complexity $H(x)$ is not computable because of the existence of the Halting problem and it is defined through a optimization process. Nevertheless, this is no obstacle to produce good and rigorous upper bounds that are enough to quantify the complexity of programs, data etc.

The classical Chaitin model is characterized by a triplet of elements $\{\mathfrak{O},\mathfrak{M},\mathfrak{F}\}$, whose definitions are:

i/ **Living Organism** $\mathfrak{O}$: it is a classical program, i.e., a piece of software that can be fed in a universal Turing Machine and produce a certain output, or just halt or even not halt. If the program $\mathfrak{O}$ halts, then the output is a string of classical bits x. In the theory of classical computation, a program $\mathfrak{O}$ can also be characterized by a certain bit-string whose size is denoted as $|\mathfrak{O}|$. Thus, $\mathfrak{O}\in\mathfrak{X}$.

The rationale behind this choice is an abstract process that reduces an organism to pure information encoded in its DNA. The rest of the organism such as its body, functionalities etc are disregarded as far as being essential to evolution is concerned. This is an oversimplification that is inherent to this toy model and so far it is necessary in order to be able to apply tools from classical information theory (AIT).

ii/ **Mutation** $\mathfrak{M}$: it is a classical algorithm that transforms a given organism $\mathfrak{O}$ into a mutated organism $\mathfrak{O}':=\mathfrak{M}(\mathfrak{O})$. Thus, it represents a transformation of the DNA by the action of external agents to the classical code. Thus, $\mathfrak{M}:\mathfrak{X}\to\mathfrak{X}$.

This notion of mutation is an algorithmic mutation as opposed to other more typical mutations called point-wise mutation that are common to population genetics studies. What is remarkable is that an algorithmic mutation is far richer than other notions of mutations considered thus far, and in this context, it appears as the most general change that we can consider on a given living organism (classical code).

Consider the following two very different mutations acting on a n-string in bitwise notation $x=x_1x_2\ldots x_n\in\mathfrak{X}$. One is a point-wise mutation $\mathfrak{P}_{no}$ defined as

$$\mathfrak{P}_{no}: x_1x_2\ldots x_{no}\ldots x_n\to x_1x_2\ldots x_{no}\oplus 1\ldots x_n, \tag{2}$$

and the other is a bit-wise mutation $\mathfrak{B}$

$$\mathfrak{B}: x_1x_2\ldots x_n\to x_1\oplus 1x_2\oplus 1\ldots x_n\oplus 1. \tag{3}$$

While $\mathfrak{P}_{no}$ represents a local change in the classical code (DNA), $\mathfrak{B}$ affects globlally to a all the code. $\mathfrak{P}_{no}$ is a typical mutation in population genetics since it is more likely to change one single base of the genetic code than multiple changes which are exponentially unlikely. On the contrary, the bit-wise mutation produces a drastic change in the genetic code. It turns out to be useful since it may lead to a change of specie for example. Both mutations are necessary and they find a common framework in the algorithmic treatment of evolution. They have similar amounts of complexity $H\left(\mathfrak{P}_{no}\right)=O(\log(n))$ and

$H(\mathfrak{B}) \leq c$. In fact, for conditional complexities we also have $H\left(\mathfrak{P}_{no}|n\right) \leq c$. Therefore, having a big mutation is not penalized during the whole history of evolution.

The evolution is a process that starts with the simplest organism $\mathfrak{O}_1$ and it evolves towards more complex organisms $\mathfrak{O}_N$ after the action of a series of mutations $\mathfrak{M}_k$, $k = 1, 2, \ldots, N$. The algorithmic complexity $H(\mathfrak{O}_k)$ measures how the new successful organisms are becoming more advanced.

It is the action of a mutation what defines the notion of time in this model and it is given by the time-step k. The total evolution time would be N.

iii/ **Fitness Function** $\mathfrak{F}$: this is a cost function that evaluates whether a mutated organism has improved with respect to the original. Thus, $\mathfrak{F}$:$\mathfrak{X} \rightarrow$R.

Let $\mathfrak{O}_k$ be a given organism and time-step. Then, in the next step the organism is mutated to $\mathfrak{O}'_k := \mathfrak{M}_k(\mathfrak{O}_k)$. The fitness function selects whether the new organism survives or fails:

$$\mathfrak{O}_{k+1} := \begin{cases} \mathfrak{O}'_k & \text{if} \quad \mathfrak{F}(\mathfrak{O}'_k) > \mathfrak{F}(\mathfrak{O}_k); \\ \mathfrak{O}_k & \text{if} \quad \mathfrak{F}(\mathfrak{O}'_k) \leq (\mathfrak{O}_k). \end{cases}$$

Chaitin's deep insight into the problem of biological evolution is the choice of the fitness function from AIT. The idea is to see life as evolving software, such that a living organism is tested after a mutation has occurred. The idea is to use a testing function that is an endless resource. This way, evolution will never be exhausted, will ever go on. In AIT there are several functions with this remarkable property that make them specially well-suited for this task: quantities that are definable but not computable. One example is the Busy Beaver function[16] $\sum$. Another example is Chaitin's Ω number[6,17,18] that represents the halting probability of self-delimiting TMs.

For the Busy Beaver function $\sum$ there are several variants which are equally good for the purposes of fitness function, that measures the rate of evolution. For instance, $\sum$ can be defined as the maximum number of 1′s output by a TM U after it halts starting from a blank input data $q = \Lambda$. To work with $\sum$ it is convenient to specify the maximum size N that the programs $p \in \mathfrak{X}$ operated by U and define the output as the largest integer $k \in \mathfrak{X}$ in binary form that is computed after halting U. Thus, a N-th Busy Beaver function is denoted $\sum_N$ and defined

$$\sum\nolimits_N := \max \quad k, \qquad H(k) \leq N \tag{5}$$

where the algorithmic complexity (1) is defined for programs p that compute $k = U(p)$ without input and halting. This is a well-defined function $\sum_N$: N → N but it is noncomputable: it grows faster than any computable function $f(N)$, $\sum_N > f(N)$ for sufficiently large N. Therefore, $\sum_N$ cannot be bounded in the form of $\sum_N = O(f(N))$. This is the property that makes $\sum_N$ a good candidate for fitness function since it is an endless source of creativity that enable us to test a new organism, a program $\mathfrak{O}$, and see whether it is smarter by checking whether it can name a bigger number. Thus, we can use (4) with $\mathfrak{F} = \sum_N$ and ask how the total mutation time T_N behaves as N grows. Let us mention in passing that naming increasingly bigger numbers requires lots of creativity in the form of new functions and ways to name new numbers bigger and bigger.

A more manageable and systematic choice for fitness function is Chaitin's Ω number. To define it, it is convenient to introduce the notion of universal probability $P_U(x)$ of a given string $x \in \mathfrak{X}$:

$$P_U(x) := \sum_{p:U(p)=x} 2^{-|p|}, \tag{6}$$

which is the probability that a program randomly drawn as a sequence of fair coin flips $p = p_1p_2\ldots$ will compute the string x. That this is a well-defined probability distribution is a central result in AIT. It relies on some technical details: a) the programs p are not arbitrary, but self-delimiting; b) convergence of the series is guaranteed by the Kraft inequality[19]. A self-delimiting program is a program that knows when to stop by itself, without additional stopping symbols. It is constructed from a set of prefix-free strings of bits: strings that are not prefix of any other string in the set (see Methods section III). In AIT, the algorithmic complexity and the universal probability of strings are related by a Shannon type of equation:

$$H(x) = -\log P_U(x) + O(1). \tag{7}$$

The Ω number can be defined from the universal probability once we drop any reference to any particular output string:

$$\Omega := \sum_{p:U(p)=\text{halts}} 2^{-|p|}, \tag{8}$$

It is considered as the halting probability in the theory of TMs. It measures the probability that a randomly chosen program p will halt when run in a UTM that halts. Thus, it is defined on the set of prefix-free halting programs, not for arbitrary programs. Interestingly enough, Chaitin proved that universal TM exist for self-delimiting programs. This technical condition guarantees that $0 < \Omega < 1$: there are always programs that halt, but not all of them will halt due to the halting problem. Again, Ω is well-defined and noncomputable. It hosts an inexhaustible amount of knowledge and it is thus suited for a fitness function. In short, if Ω were computable it would imply that there is no halting problem, which is false. Like $\sum$, it is convenient to truncate Chaitin's number up to programs of size N computed in time less than N:

$$\Omega_N := \sum_{p:|p|<N, T<N} 2^{-|p|}. \tag{9}$$

These Ω_N are lower bounds to the actual Ω. This truncation also produces an unbounded function Ω_N that reflects its non-computability.

Chaitin uses Ω_k to define an organism $\mathfrak{O}_k$ and a mutation $\mathfrak{M}_k$ at time-step k, as well as the fitness function $\mathfrak{F}$. Namely, an organism is defined by means of the first $N(k)$ binary digits ω_i of Ω_k:

$$\mathfrak{o}_k := \omega_1\omega_2\ldots\omega_N(k). \tag{10}$$

To complete the construction of the organism $\mathfrak{O}_k$ from the proto-organism $\mathfrak{o}_k$, we need two more ingredients. One is to make it a self-delimiting program by including a prefix string $1^{N(k)}0$ (see Methods section III) and the other one is to prefix a program p_Ω to read the fitness of the resulting organism. Altogether, the organism looks like:

$$\mathfrak{O}_k := p_\Omega\ 1^{N(k)}0\ \mathfrak{o}_k. \tag{11}$$

The mutation acts on the organism by trying to improve the lower bounds on Ω. According to AIT, a natural move is

$$\mathfrak{M}_k : \Omega_k \rightarrow \Omega'_k = \Omega_k + \frac{1}{2^k}. \tag{12}$$

Notice that this mutation induces, in turn, a mutation in the organism $\mathfrak{O}_k \rightarrow \mathfrak{O}'_k$ by the rules specified in its construction above. These mutations represent challenging an organism to find a better a better lower bound of Ω which amounts to an ever increasing source of knowledge. To this end, the fitness function $\mathfrak{F} = \Omega$ is introduced as follows:

$$\mathfrak{O}_{k+1} := \begin{cases} \mathfrak{O}'_k & \text{if} \quad \Omega'_k < \Omega; \\ \mathfrak{O}_k & \text{if} \quad \Omega'_k \geq \Omega. \end{cases} \tag{13}$$

To understand this selection, notice that no truncation Ω_k can be greater that the real Ω and thus, this represents a failure. On the contrary, if the new truncation Ω'_k is still less than Ω, we have increased our knowledge of how many programs will halt upon running a UTM ($\Omega'_k > \Omega_k$). As Chaitin notices, this implies the use of an oracle[2–4].

It is possible to define a variant of the Busy Beaver function $\tilde{\sum}_k$ in terms of Ω_N as the least N for which the first k bits of the binary string of Ω_N are correct. In AIT it can be proved that both Busy Beavers are approximately equal,

$$\tilde{\sum}_N = \sum_{N+O(log(N))}. \tag{14}$$

B. Chaitin's Evolution Scenarios. Let us denote T_N the total mutation time, i.e., the number of mutations tried in order to evolve an initial organism $\mathfrak{O}_1$ up to a certain more fitted organism $\mathfrak{O}_N$. Depending on the strategy followed by Nature, Chaitin considers three scenarios and computes the scaling of T_N with N. In this way, one can assess which is the best evolutionary scenario. The results are the following:

- Scenario I: *Exhaustive Search.*

This scenario represents that there is no strategy in Nature and every possible organism is tested regardless which was the previous organism that originated it. Thus, there is no effective application of a fitness function but Nature explores all possible codes available in the phase space. As from AIT we know that in a given set of strings $\mathfrak{X}_N$ of length up to N there are $2^N - 1$ strings, then the order of the evolution time is

$$T_N = O\left(2^N\right). \tag{15}$$

It takes an exponential time to reach a certain organism $\mathfrak{O}_N$.

- Scenario II: *Intelligent Design.*

This scenario is the opposite to the previous one. Now, Nature is not dumb but assumed to be intelligent enough so as to know about AIT and this model of evolution. The initial proto-organism is $\mathfrak{o}_1 = 0$. The best strategy is to apply a process of interval halving to track down better lower bounds to O by applying mutations $\mathfrak{M}_k$, $k = 1, 2, \ldots, N$ in this increasing order. Thus the mutation time takes of the order of N trials:

$$T_N = O(N) \tag{16}$$

Thus, by selecting intelligently the order of the mutations, since we assume that Natures knows the structure of Ω, then the total evolution time for an organism grows linearly in N.

- Scenario III: *Cumulative Evolution at Random.*

A more natural assumption is that Nature choses randomly the mutations $\mathfrak{M}_k$ among the set of possible mutations. It is a random walk in the space of mutations. Remarkably enough, the evolution time grows in between quadratic and cubic in N:

$$T_N = O\left(N^{2+\delta}\right), \quad 0 < \delta < 1. \tag{17}$$

Although this is worse than scenario II, it is still a polynomial growth and far from the exponential growth of scenario I.

C. Quantum Chaitin Model. The following definitions are we well-motivated when trying to bring concepts from Quantum Information Theory (QIT) into Chaitin's classical model. They can be made even more general as discussed in Sect.II.

i/ **Quantum Organism** $\mathfrak{O}^q$: it is a pure quantum state in a Hilbert space $\mathcal{H}$ of infinitely countable qubits: $\mathfrak{O}^q := |\Psi\rangle \in \mathcal{H}$. In practice, we shall be dealing with a finite truncation to a number of qubits N denoted as $\mathcal{H}_N$.

The meaning of this choice is motivated by the notion of classical organism as a program for a TM. Now, the quantum version is a pure state that encodes the information of a quantum program. This is meaningful since we have adhered to an abstraction process in which a living organism is divested of everything except its genetic code that is represented by a classical program. Thus, a quantum organism is not a form of quantum life, but represents quantum effects in the classical code of DNA.

ii/ **Quantum Mutation** $\mathfrak{M}^q$: it is a quantum algorithm that transforms the original quantum organism $\mathfrak{O}^q$ into a mutated quantum organism $\mathfrak{O}'^q$:

$$\mathfrak{M}^q : \mathfrak{O}^q \rightarrow \mathfrak{O}'^q. \tag{18}$$

iii/ **Quantum Fitness** $\mathfrak{F}^q$: it is a cost function that selects a mutated organism when it is fittest than the original.

The traditional characters of Quantum Information[20,21] Alice A and Bob B, can be adapted to the quantum evolution scenario: Alice is the organism before the mutation $A = \mathfrak{O}^q$ and Bob is the mutated organism $B = \mathfrak{O}'^q$. Then, $\mathfrak{M}^q$ will success or fail depending on the fitness of the pair (A, B).

In order to complete the above quantum definitions we need to specify how to choose a triplet $\{\mathfrak{O}^q, \mathfrak{M}^q, \mathfrak{F}^q\}$ in the quantum case. We shall follow the classical model and try to find a quantum version of organisms as lower bounds to some Ω number to be specify. Once this is done, the quantum notions of mutation and fitness function will also follow. All this can be done by defining a notion of quantum algorithmic complexity.

D. Quantum Algorithmic Complexity. The quantumness of the Ω number that we are searching for our definition of quantum organism will depend on the notion of quantum algorithmic complexity H_q that we decide to use. In fact, there are several versions of H_q[22–25] and not all of them are equivalent. We shall choose the definition of Mora and Briegel[25] that is called network complexity H_{net} because of the following properties[25–27]:

a/ H_{net} is a classical algorithmic complexity associated to a quantum state. It describes how many classical bits of information are required to describe a quantum state of N qubits. Being classical, it will allow us to compare to previous evolution rates on equal footing.

b/ H_{net} has the special property that it requires an exponential number of classical bits for the description of generic quantum states. In particular, it detects a sharp difference between multipartite entangled states and separable states.

The network complexity is a description that Alice does of a quantum state $|\Psi\rangle_N$ she has and she wants to send this information to Bob through a classical channel so that Bob could eventually reproduce that state on his side. It describes the classical effort Bob would have to do. In order to define network complexity, we need several operational elements: a) a universal set of quantum gates $\mathcal{S}$; b) an alphabet to code circuit operations $\mathcal{A}$, and c) a fidelity or degree of precision $\epsilon \in (0,1)$. With the aid of these elements, we can construct a mapping from quantum states in $\mathcal{H}_N$ to finite strings $\mathfrak{X}$, such that

$$\mathfrak{O}_{\text{cl}} : |\Psi\rangle_N \rightarrow x_\psi, \tag{19}$$

and then,

$$H_q(|\Psi\rangle) = H_{\text{net}}(|\Psi\rangle) := H(x_\psi). \tag{20}$$

The first equality represents our choice of quantum algorithmic complexity while the second is the definition of network complexity (1).

The mapping $\mathfrak{O}_{\text{cl}}$ (19) is constructed from the elements a)-c) as follows: let us select a universal finite set of gates for example, the one generated by the gates $\mathcal{S} = \{U_{\text{H}}, U_{\text{K}}, U_{\text{Cnot}}\}$[28], i.e., the Hadamard gate, the $\pi/8$-phase gate and the Cnot gate, respectively. Then, Alice sets up a quantum circuit of gates called U by concatenating gates from $\mathcal{S}$, and constructs a state, namely, $U|0\rangle_N$, from an initialization state $|0\rangle_N := |0\rangle^{\otimes N}$. This prepared state can approximate the desired state $|\Psi\rangle_N$ with precision given by

$${}_N\langle\Psi|U|0\rangle_N \geq 1 - \epsilon. \tag{21}$$

In all what follows, ϵ will be a fixed parameter once and for all from the beginning.

Next, Alice needs to use the alphabet $\mathcal{A}$ in order to code all the operations in the circuit U and preparation of the state with ϵ-precision (21). This is represented by a certain string of bits $\mathcal{A}(U,\epsilon) := \alpha_1\alpha_2\ldots\alpha_M$, where M is the length of the resulting bit-string and is a certain function of the number of qubits N. Then, the mapping (19) is given by

$$\mathfrak{O}_{cl}(|\Psi\rangle_N) = x_\Psi := \mathcal{A}(U,\epsilon). \tag{22}$$

With this, the network complexity (20) is well-defined. An additional minimization process is assumed in (1) since the circuit U is not unique and it is natural to request to use the minimal circuit that prepares the state with the desired precission.

Our choice of quantum algorithmic complexity has very important consequences for studying quantum effects in biological evolution:

1/ According to this definition of quantum algorithmic complexity in terms of a classical network complexity, we realize that the set of quantum states is mapped onto the set of bit-strings. Thus, while the former is uncountable, the latter is infinitely denumerable.
2/ By virtue of this mapping we are complying with the Turing barrier.
3/ The fact that the network complexity is classical will make that our quantum Ω^q will be also real numbers and not quantum states or operators. However, we can make classical definitions of Ω numbers that represent different types of quantum states (see later).
4/ In a traditional quantum information scenario, Bob needs to agree with Alice on which alphabet to use in order to communicate. In a quantum evolution scenario, there is no need to agree on a common language for the description since there are not two observers, but a single organism that evolves.

We shall use the following fundamental results from network complexity and quantum states[25]. As a consequence of the Solovay-Kitaev theorem[20,29,30], the number of gates (bit-string) M of the circuit needed to construct a given multipartite state $|\Psi\rangle_N$ grows exponentially with N for a fixed accuracy ϵ.

$$H_{net}(|\Psi\rangle_N) \lesssim 2^N \log\frac{1}{\epsilon}. \tag{23}$$

Furthermore, the network complexity quantifies very differently the complexity of separable and entangled states[25]:

- Separable States $|\Psi\rangle_S$:

$$H_{net}(|\Psi\rangle_S) \lesssim N\ \log\frac{1}{\epsilon}. \tag{24}$$

- Maximally Entangled States $|\Psi\rangle_E$:

$$H_{net}(|\Psi\rangle_S) \lesssim N\ 2^N\ \log\frac{1}{\epsilon}. \tag{25}$$

- Generic States:

$$H_{net}(|\Psi\rangle) \lesssim N\ 2^{E_s(|\Psi\rangle)}\ \log\frac{1}{\epsilon}. \tag{26}$$

where $E_s(|\Psi\rangle)$ is the Schmidt measure which quantifies the degree of entanglement in the multipartite pure state[31].

The fact that separable states are less complex than entangled states means that separable states are more likely: If we type a random bit-string at a computer, most likely it will correspond to a separable state. This raises a fundamental question: can we use the higher complexity of entangled states to accelerate the rate of biological evolution? To answer this question we need to introduce the corresponding quantum Ω numbers and different scenarios for mutations evolution in which evolution will develop.

E. Quantum Omega Numbers. In order to describe different types of quantum organisms we need to define different types of Ω numbers associated to quantum states. Thus, we shall use the basic results on network complexity $H_{\rm net}$. However, we can define Omega numbers associated to selected classes of states. As we know from the geometry of the Hilbert space of states that the set of separable states does not intersect the set of truly entangled (maximally) states, we can define Ω numbers by restricting the sum on the programs originated by the mapping (19) to those yielding either separable or entangled states. By construction, these sets are discrete since we are using a discrete set of universal quantum gates $\mathcal{S}$.

- Separable $\tilde{\Omega}_{\rm S}$ number:

$$\tilde{\Omega}^{\rm S} := \sum_{p_{\rm S}:U(p_{\rm S})=\text{halts}} 2^{-|p_{\rm S}|}, \tag{27}$$

where $p_{\rm S}$ is a program that describes the network complexity of a separable state $|\Psi\rangle_{\rm S}$. To do this sum, we construct all possilble separable states and apply the mapping (19) to perform the sum. As the method is constructive, the separable states are obtained on demand.

- Entangled (maximally) $\tilde{\Omega}_{\rm E}$ number:

$$\tilde{\Omega}^{\rm E} := \sum_{p_{\rm E}:U(p_{\rm E})=\text{halts}} 2^{-|p_{\rm E}|}, \tag{28}$$

where $p_{\rm E}$ is a program that describes the network complexity of a maximally entangled state $|\Psi\rangle_{\rm E}$. To do this sum, we fix the accuracy ϵ which behaves as an overhead factor, then we construct all posilble maximally entangled states and apply the mapping (19) to perform the sum. The decision problem of whether a given constructed state is maximally entangled is solved by computing its Schmidt measure and testing that it is maximal. We take this as an operational definition of maximally entangled state in this context.

In both sums, the programs $p_{\rm S}$ and $p_{\rm E}$ are assumed to be prefix-free in order to guarantee their convergence. The typical behaviour of their general terms are 2^N and 2^{N2^N}, repectively. We drop off the overhead factor from now on. From the viewpoint of AIT, we may use another equivalent definitions in terms of the network complexity explicitly:

$$\Omega^{\rm S} := \sum_{x^S:U(x^S)=\text{halts}} 2^{-H_{\rm net}(|\Psi\rangle^S)}, \tag{29}$$

$$\Omega^{\rm E} := \sum_{x^E:U(x^E)=\text{halts}} 2^{-H_{\rm net}(|\Psi\rangle^E)}. \tag{30}$$

The above quantum Ω numbers are introduced relying on the choice of quantum algorithmic complexity in terms of network complexity. Other choices of quantum complexity may lead to different definitions of quantum Ω numbers that may become quantum states[32,33] or even quantum operators.

F. Quantum Evolution Scenarios. We want to compare quantum evolution in a world of maximally entangled quantum organisms w.r.t. a classical world both in intelligent design and cumulative evolution scenarios.

In order to study quantum effects in evolution scenarios as in Sect.I A, (15) (16) (17), we need to define a triplet $\{\mathfrak{O}_k^q, \mathfrak{M}_k^q, \mathfrak{F}_k^q\}$. This is achieved by introducing truncated versions of the quantum Ω numbers in (29), as follows. For separable states, we have

$$\Omega_N^{\mathrm{S}} := \sum_{n<N} 2^{-H_{\mathrm{net}}\left(|\Psi\rangle_n^{\mathrm{S}}\right)}, \tag{31}$$

where the sum runs over truncations up to N qubits, $|\Psi\rangle_n^{\mathrm{S}} \in \mathcal{H}_n$ $n=1,2,\ldots,N-1$, corresponding to the construction process described in (27), (29). The quantum separable organism is a lower bound to (31). The key distinctive feature is that the typical behaviour of one element in this truncated sum decreases as 2^{-k}. Thus, the corresponding mutation is defined such as to produce a significant change in the organisms as

$$\mathfrak{M}_k^q : \Omega_k^{\mathrm{S}} \to \Omega'^{\mathrm{S}}_k = \Omega_k^{\mathrm{S}} + \frac{1}{2^k}. \tag{32}$$

Therefore, the analysis of the evolution rates for the quantum evolution scenarios dealing with separable states are similar to those with classical organisms in (15), (16), (17). A classical state is a state that can be prepared classically, thus it can evolve classically. The same treatment as with the classical scenarios in Sect.I A reproduces the same evolution rates.

A different result will be obtained with maximally entangled organisms. Now, let us introduce the truncated entangled Ω number as

$$\Omega_N^{\mathrm{E}} := \sum_{n<N} 2^{-H_{\mathrm{net}}\left(|\Psi\rangle_n^{\mathrm{E}}\right)}, \tag{33}$$

This allows us to obtain a quantum version of the triplet $\{\mathfrak{O}_k^q, \mathfrak{M}_k^q, \mathfrak{F}_k^q\}$. In particular, the quantum entangled organism $\mathfrak{O}_k^q$ at time step k is defined by the same process in Sect.I A, (11) of producing lower bounds but now with the truncated quantum Ω number (33). $\mathfrak{O}_k^q$ yields a lower bound to Ω_N^{E} defined above (33).

Next, we introduce a mutation $\mathfrak{M}_k^q$ that tries to make this quantum organism $\mathfrak{O}_k^q$ to progress. A significative progress will occur if we try to increase the form of the quantum Ω number (33) according to the typical behaviour of its terms in the sum. This is given by 2^{-N2^N} for spaces up to N qubits. Thus, now we define an entangled version of the mutation as

$$\mathfrak{M}_k^q : \Omega_k^{\mathrm{E}} \to \Omega'^{\mathrm{E}}_k = \Omega_k^{\mathrm{E}} + \frac{1}{2^{k2^k}}. \tag{34}$$

Notice that this choice of move in the space of quantum organisms is motivated by the typical behaviour of quantum circuits representing quantum algorithms acting on quantum states. This is the natural scale for quantum mutations to occur at the level of quantum organisms.

The fitness function is determined by the oracle of Ω^{E} which decides whether the mutated organism $\mathfrak{O}'^q_k$ with (34) succeeds or fails according to the criteria (13).

Now, we have all the ingredients to analyze the rates for different quantum evolution scenarios, mainly with entangled organisms.

1. Quantum Exhaustive Search. As indicated by its name, this strategy is defined by searching all classical possible programs that can be generated quantum states available in the Hilbert space $\mathcal{H}_N$ of N qubits by means of the mapping (19). For a strings of length M we know this grows as 2^M. In turn, the length of these strings is related to the number of qubits as $M \approx 2^N$. Thus, as in this evolution scenario each mutation is exhaustive, i.e., it tries every possible quantum organism regardless which original organism may be, then the evolution rate behaves as

$$T_N = O\left(2^{2^N}\right). \tag{35}$$

2. Quantum Intelligent Design. This strategy is like climbing a hill via the optimal path, knowing such a path before hand. In such a way that each step is always better than the previous one. There is no backtracking. A more proper name would be Quantum Optimal Evolution.

Now, we have to use the quantum mutations (34). If we produce an optimal ordered sequence of these mutations M_k^q as follows: $k = 1, 2, \ldots, N$ we shall reach the first N valid digits of Ω_N^{E}, by construction, and then the evolution rate is:

$$T_N = O(N) \tag{36}$$

Thus, quantum intelligent design behaves linear in the number of trials N in a maximally entangled world. This behaviour is equal as the intelligent design in a classical world (16). Notice that the quantum mutations have a different growth rate than classical mutations, but nevertheless the evolution time is the same: they are optimal.

3. Quantum Cumulative Evolution. This strategy is like climbing a hill, but now we do not have a priori knowledge of the best strategy to improve the lower bounds of the quantum number. Thus, a natural strategy is to mutate by means of a random walk in the space of quantum mutations given by (34). In this case, the quantum mutations must be drawn at random and often enough so as to produce the same final quantum organism.

The quantum mutation is characterized by the growth $k2^k$. For simplicity, we shall take it as the leading behaviour 2^k. As the we have chosen the network complexity as our measure for quantum algorithmic complexity, we can now use classical formulas and Methods section III, (45) to estimate the complexity of a quantum mutation associated to a maximally entangled state:

$$H\left(\mathfrak{M}_k^q\right) \lesssim \log 2^k + 2\log\log 2^k + O(1) \approx k + 2\log k. \tag{37}$$

Its probability is $2^{-H\left(2^k\right)} = 1/\left(2^k k^2\right)$ and its frequency is $k^2 2^k$. The total evolution time T_N is of the order of

$$T_N = \sum_{k=1,\ldots,N} k^2 2^k, \tag{38}$$

which grows exponentially up to polynomial factors,

$$T_N = O\left(2^N\right). \tag{39}$$

Thus, quantum cumulative evolution in a maximally entangled world behaves exponentially worse than cumulative evolution in a classical world. Quite likely, it is more favorable to evolve in a classical world than in a quantum world. This may explain why we live in a classical world at the macroscopic level. We should remark that this conclusion does not contradicts the fact that quantum algorithms can be more efficient than classical algorithms since our conclusions refer to algorithmic complexity, while quantum algorithms deal with computational complexity (time and space resources for computation).

Discussion

We have studied quantum effects on biological evolution by means of a descriptive toy model based on quantum algorithmic complexity. This is an adequate option when studying biological evolution from a broad perspective and in a very large time scale, so large that any type of quantum mutation (18) can take place and not just point-wise mutation that only affect a base in the DNA code. In quantum evolution terms, the quantum complexity is a measure of how difficult has been for Nature to 'prepare' the quantum organism. The results obtained in Sect.I F for the rates in quantum evolution scenarios are based not on the notion of runtime complexity, but on the notion of mutation time, as well as what a typical quantum mutation move is.

The halting problem and other noncomputable functions are preceded by an aura of being a pathology, a nuisance … eventually, something negative. This is the perspective of non-specialists. On the contrary, we may consider this undecidability as a sort of intrinsic randomness in Mathematics. This is analogous to the intrinsic

randomness that quantum theory brought to Physics even earlier in the history of Science. Now we know from Quantum Information Theory (QIT) that this randomness can be used to our benefit, in a large variety of ways. Similarly, It is very remarkable how Chaitin turns the problem of non-computability in algorithmic complexity into a source of creativity in order to challenge living organisms to evolve by becoming increasingly more advanced. This process of challenging by means of mutations is endless precisely because the fitness function employed is non-computable and cannot be bounded when truncated, as we learn from AIT. Thus, non-computability is given a positive role in a descriptive version of biological evolution.

We have adopted the same perspective when formulating a quantum version of an algorithmic model for biological evolution. This has motivated us to use a quantum notion of complexity based on the network complexity. In this way, We can still work with lower bounds of quantum Ω numbers as prototype of quantum effects in DNA code. This perspective is non-trivial in the quantum case since it implicitly assumes the existence of the Turing barrier also in the quantum realm. This is still an open problem. While a classical Turing Machine works with data and programs that are infinite but countable, a quantum Turing Machine works with non-countable sets like complex numbers. Thus, we could argue that the classical halting problem does not apply since now the number of quantum TMs is uncountable. However, let us recall that Turing's halting problem is just one instance, very remarkable, of Gödel's incompleteness theorem. Thus, if we believe that Gödel's incompleteness theorem applies beyond Arithmetic, we may accept that there are uncomputable problems in quantum TMs also, and likely something equivalent to a quantum Turing halting problem. Moreover, in our case, we have employed a finite set of universal quantum gates and a transformation to bit-strings from network complexity. This implies that we are not considering a quantum TM as a continuous system, but we are dealing effectively with a countable set of gates up to a fixed precision ϵ. T his seems the simplest generalization of the classical scenario.

It is interesting to realize that the Turing barrier has important consequences in a quantum evolution scenario of this kind. In case that barrier could be beaten by quantum effects, that would imply that we cannot use real quantum non-computability as a source of creativity in quantum evolution as in the classical Chaitin model. We could not justify quantum effects on biological evolution on the same theoretical grounds.

The evolution rates in quantum scenarios are understood up to an overhead factor arising from the accuracy factor that we want to use. This is fixed and thus removed from the expressions for simplicity. However, this ϵ parameter is new in the quantum evolution case and does not exist in the classical case. Something similar could be introduced in a classical evolution by invoking the existence of classical errors during evolution, but this is not standard in AIT. The reason for the existence of ϵ is because the universal gate set $\mathcal{S}$ is finite. With a continuous universal gate set[20,21,34] it is possible to get rid of it, but that would imply that Nature would had an infinite amount of resources, something which we do not consider reasonable. The fundamental origin of this difference is the fact that the set of classical strings is countable while the set of quantum states is uncountable. Thus, working with a classical universal set of gates does not need an ϵ parameter. In this regard, the quantum complexity is more 'natural' than the classical where something like ϵ is absent at the very fundamental level. In other words, the classical universal Turing machine and the finite universal quantum gate set are not on equal footing, but the quantum case is more 'natural' since Nature can also make errors.

Alternatively, we can think of this parameter as a grid or lattice spacing but in the space of quantum states, rather than in real space. It is a discretization. In this sense, we always work with a finite lattice or grid, and that is why we drop this dependence. We never take the continuum limit.

The network complexity is formulated in terms of a finite universal gate set $\mathcal{S}$, instead of a quantum Turing Machine which would seem more natural if we see how the classical algorithmic complexity is defined explicitly in terms of a classical Turing Machine. However, this is not an obstacle since we are using the Solovay-Kitaev theorem to reconstruct arbitrary quantum unitary gate to a given precision. Furthermore, we also know that the quantum circuit model is equivalent to the quantum Turing Machine model due to the Yao theorem[35]. Moreover, we have also identified that the choice of a coding language in network complexity to transform a quantum circuit in a bit-string (19) is irrelevant for quantum evolution, since Alice and Bob are replaced by the original organism and the mutated organism, respectively.

The simple quantum toy model of Sec.I C can be thought of as a first step towards more realistic models and it does not exhaust all necessary ingredients to describe quantum effects in biological evolution, even from a algorithmic information viewpoint. For example, we can mention a series of extensions that this model still allows:

Fitness Functions: instead of using lower bounds to quantum Ω numbers, there are other options considered by Chaitin in his classical model that it is worthwhile to find its instance in the quantum model.

Creation of Hierarchies: classical evolution favors the formation of hierarchies of living organisms. Are they compatible with quantum evolutions?

Mixed States: in all our analysis, quantum organisms have been represented by means of pure quantum states and mutations by quantum algorithms. There is a natural extension to mixed quantum states where the lack of purity here may represent the action of an external environment on the organism, i.e., its genetic code. This degree of freedom may influence the evolution rates and it is also a way to model the presence of errors during evolution.

Continuous Variables (CV): we may also choose the Hilbert space of states not to be represented by qubits, but for continuous variable states[36], like Gaussian states. This is still a well-defined model for quantum computation and it remains a challenge to study its properties from the algorithmic complexity perspective and in the context of evolution.

Quantum Complexity: as we have mentioned, there are other notions of quantum algorithmic complexity that are not equivalent to network complexity. It is still possible to keep a version of this toy model in Sect.I C and investigate the consequences in quantum evolution of these other choices. In particular, quantum Ω numbers can be replaced by quantum states or even quantum operators. This may affect the evolution rates. However, it is important to justify conceptually these other choices.

Methods

Size of Self-Delimiting Bit-Strings in Algorithmic Complexity Theory (AIT). The size of an integer $k \in \mathrm{N}$ is defined as

$$\mathrm{size}(k) := 1 + \lceil \log(1+x) \rceil, \tag{40}$$

where $x \in \mathfrak{X}$ is the finite bit-string representation of k.

In AIT, it is technically necessary to work with self-delimiting, i.e., prefix-free strings of bits in order to have a well-defined halting probability Ω that be convergent. Let x denote a n-bit string: $x = x_1 x_2 \ldots x_n$. The set of $x \in \mathfrak{X}$ strings is not self-delimiting. For example, for $n = 2$ then $\mathfrak{X} = \{0, 1, 00, 01, 10, 11\}$ has a 0 that is a prefix of 00 and 01 and so on.

Given a set of strings $\mathfrak{X}$ we can make them into a set of self-delimiting strings by the following procedure:

$$x \mapsto 0^n 1\, x =: \mathfrak{O}_x \tag{41}$$

i.e., we put n 0s before the string and use a 1 to separate it from the given n-bit string x. In the context of evolution, this is called a proto-organism. As the Turing Machine has to read the input string bit by bit, then this way we are telling the TM the length of the string x ahead of time (before the TM reads it). Another example for $n = 3$ bit-strings is

$$0\mapsto 010,\ \ 00\mapsto 00100,\ \ 10\mapsto 00110,$$
$$1\mapsto 011,\ \ 01\mapsto 00101.\ \ 11\mapsto 00111, \tag{42}$$

and now we do not have prefix strings anymore.

The size of self-delimiting strings enters in the definition of the Chaitin numbers and we need to compute its size. Denote $|x|$ the size in bits of a string $x \in \mathfrak{X}$. In our case, $|x| = n$. Then, the size of its self-delimiting extension (41) is

$$|\mathfrak{O}_x| = 2n+1. \tag{43}$$

However, we realize that the coding of the size of the string x in the above self-delimiting procedure is highly inefficient since we are using the unary code. We can improve this coding by using the fact that an integer like $|x|$ can be coded with $\log n$ bits, for x large (40). Thus, let us define a better encoded self-delimiting string x' as follows

$$x' := |\mathfrak{O}_x|_{\text{red}}\ x, \tag{44}$$

where $|\mathfrak{O}_x|_{\text{red}}$ here is the string of bits representing the log of the size of $\mathfrak{O}_x$, and appears before x. Now, its size is (43)

$$|x'| = n + 2\log n + 1. \tag{45}$$

1. Darwin, Ch. *On the Origin of Species by Means of Natural Selection, or the Preservation of Favoured Races in the Struggle for Life*, (1959).
2. Chaitin, G. J. To a mathematical theory of evolution and biological creativity; preprint (2011). Paper presented Monday 10 January 2011 at a workshop on √íRandomness, Structureand Causality: Measures of Complexity from Theory to Applications √íorganized by Jim Crutchel.
3. Chaitin, G. J. *Proving Darwin*, Pantheon, New York, (2012).
4. Chaitn, G. J. Life as evolving software, in H. Zenil, *A Computable Universe*, World Scientific, Singapore, (2012).
5. Chaitin, G. J. A theory of program size formally identical to information theory. *J. ACM* **22**, 329, –340 (1975).
6. Chaitin, G. J. *Algorithmic Information Theory*, Cambridge University Press, (1987).
7. Turing, A. M. On computable numbers, with an application to the Entscheidungsproblem, *Proc. London Math. Soc.* **42**, 230–265 (1936–37); Correction, ibid. **43**, 544–546 (1937).
8. Feynman, R. P., Simulating physics with computers. *Int. J. Theor. Phys.* **21**, 467 – 488 (1982).
9. Bernstein, E. & Vazirani, U. Quantum complexity theory. *SIAM J. Comput.* **26**, 1411 (1997).
10. Calude, C. S & Pavlov, B. Coins, Quantum Measurements, and Turing √ísBarrier, *Quantum Information Processing*, **1**, *Nos*.1/2, (2002).
11. Kieu, T. D. Quantum Algorithm for Hilbert√ísTenthProblem, *arXiv:quant-ph/0310052v2* (2003).
12. Schrödinger, E. *What Is Life? The Physical Aspect of the Living Cell.* Cambridge University Press, Cambridge (1944).
13. Solomonoff, R. A Formal Theory of Inductive Inference Part I. 7 *Information and Control* **7** (1): 1–22 (1964).
14. Kolmogorov, A. N. Three Approaches to the Quantitative Definition of Information. *Problems Inform. Transmission* **1**, 1–7 (1965).
15. Chaitin, G. J. On the length of programs for computing finite binary sequences. *Journal of the ACM* **13**, 547–569 (1966).
16. Radó, T. On non-computable functions, Bell *System Technical Journal*, **41**, No. 3, 877–884 (1962).
17. Chaitin, G. J. *The Limits of Mathematics,* Springer-Verlag, London, (2003).
18. Calude, C. C., Dinneen, M. J. & Shu, C.-K. Computing a glimpse of randomness. *Exper. Math.*, **11**, 361–370 (2002).
19. Cover, T. M. & Thomas, J. A. *Elements of Information Theory*, Second Edition, John Wiley & Sons, Inc. New Jersey (2006),
20. Nielsen, M. A. & Chuang, I. L. *Quantum Computation and Quantum Information,* Cambridge University Press, UK, (2000).
21. Galindo, A. & Martin-Delgado, M. A. Information and Computation: Classical and Quantum Aspects. *Rev. Mod. Phys.* **74**, 347–423, (2002); arXiv:quant-ph/0112105.
22. Vitanyi, P. Three Approaches to the Quantitative Definition of Information in an Individual Pure Quantum State, *arXiv: quant-ph/9907035* (2000).
23. Berthiaume, A., Dam, W. van & Laplante, S. Quantum Kolmogorov Complexity, *arXiv: quant-ph/005018* (2000).
24. Gács, P. Quantum Algorithmic Entropy, *arXiv: quantph/0011046* v2 (2001).
25. Mora, C. & Briegel, H. J. Algorithmic complexity and entanglement of quantum states, *Phys. Rev. Lett.* **95**, 20 (2005).
26. Mora, C. & Briegel, H. J. Algorithmic complexity of quantum states, *arXiv:quant-ph/0412172.* (2004).
27. Mora, C., Briegel, H. J. & Kraus, B. Quantum Kolmogorov complexity and its applications, *arXiv:quant-ph/0610109.* (2006).
28. Boykin, P. O., Mor, T., Pulver, M., Roychowdhury, & F. Vatan, F. On Universal and Fault-Tolerant Quantum Computing. *Information Processing Letters* **75**, 101 (2000).
29. Solovay, R. *unpublished manuscript* (1995).
30. Kitaev, A. Y. Quantum computations:algorithms and error correction. *Russ. Math. Surv.* **52**, 1191–1249 (1997).
31. Eisert, J. & Briegel, H. J. Schmidt measure as a tool for quantifying multiparticle entanglement. *Phys. Rev. A* **64** (2001).
32. Svozil, K. Quantum algorithmic information theory, eprint. *arXiv:quant-ph/9510005*, (1995).
33. Svozil, K. Halting probability amplitude of quantum computers. *Journal of Universal Computer Science* **1**, nr. 3 (1995)
34. Barenco, A., *et al*. Elementary gates for quantum computation. *Phys. Rev. A.* **52**, 3457. (1995).
35. Yao, A. *Proceedings of the 34th IEEE Symposium on the Foundations of Computer Science* (IEEE Computer Society, Los Alamitos, CA), p. 352. (1993).
36. Braunstein, S. L. & Pati, A. K. *(Eds) Quantum Information with Continuous Variables;* Kluwer Academic Publishers, The Netherlands (2003).

Acknowledgments

M.A.M.-D. thanks the Spanish MICINN grant FIS2009-10061, CAM research consortium QUITEMAD S2009-ESP-1594, European Commission PICC: FP7 2007-2013, Grant No. 249958, UCM-BS grant GICC-910758.

Additional information

Competing financial interests: The authors declare no competing financial interests.

How to cite this article: Martin-Delgado, M.A. On Quantum Effects in a Theory of Biological Evolution. *Sci. Rep.* **2**, 302; DOI:10.1038/srep00302 (2012).