entropy

MDPI

Article

The Thermodynamics of Network Coding, and an Algorithmic Refinement of the Principle of Maximum Entropy †

Hector Zenil [1,2,3,4,5,6,*], Narsis A. Kiani [1,2,3] and Jesper Tegnér [2,5,6]

1 Algorithmic Dynamics Lab, Karolinska Institute, 17177 Stockholm, Sweden; narsis.kiani@ki.se
2 Unit of Computational Medicine, Center for Molecular Medicine, Department of Medicine Solna, Karolinska Institute, 17177 Stockholm, Sweden; jesper.tegner@ki.se
3 Algorithmic Nature Group, Laboratory of Scientific Research (LABORES) for the Natural and Digital Sciences, 75006 Paris, France
4 Oxford Immune Algorithmics, Oxford University Innovation, Reading RG1 7TT, UK
5 Biological and Environmental Sciences and Engineering Division, King Abdullah University of Science and Technology (KAUST), Thuwal 23955, Saudi Arabia
6 Computer, Electrical and Mathematical Sciences and Engineering Division, King Abdullah University of Science and Technology (KAUST), Thuwal 23955, Saudi Arabia
* Correspondence: hector.zenil@algorithmicnaturelab.org
† An animated video explaining some of the methods is available at: https://www.youtube.com/watch?v=BEaXyDS_1Cw.

Received: 9 April 2019; Accepted: 20 May 2019; Published: 3 June 2019

Abstract: The principle of maximum entropy (Maxent) is often used to obtain prior probability distributions as a method to obtain a Gibbs measure under some restriction giving the probability that a system will be in a certain state compared to the rest of the elements in the distribution. Because classical entropy-based Maxent collapses cases confounding all distinct degrees of randomness and pseudo-randomness, here we take into consideration the generative mechanism of the systems considered in the ensemble to separate objects that may comply with the principle under some restriction and whose entropy is maximal but may be generated recursively from those that are actually algorithmically random offering a refinement to classical Maxent. We take advantage of a causal algorithmic calculus to derive a thermodynamic-like result based on how difficult it is to reprogram a computer code. Using the distinction between computable and algorithmic randomness, we quantify the cost in information loss associated with reprogramming. To illustrate this, we apply the algorithmic refinement to Maxent on graphs and introduce a Maximal Algorithmic Randomness Preferential Attachment (MARPA) Algorithm, a generalisation over previous approaches. We discuss practical implications of evaluation of network randomness. Our analysis provides insight in that the reprogrammability asymmetry appears to originate from a non-monotonic relationship to algorithmic probability. Our analysis motivates further analysis of the origin and consequences of the aforementioned asymmetries, reprogrammability, and computation.

Keywords: second law of thermodynamics; reprogrammability; algorithmic complexity; generative mechanisms; deterministic systems; algorithmic randomness; principle of maximum entropy; Maxent

---

> Thermodynamics is a funny subject. The first time you go through it, you don't understand it at all. The second time you go through it, you think you understand it, except for one or two small points. The third time you go through it, you know you don't understand it, but by that time you are so used to it, it doesn't bother you any more.
>
> Arnold Sommerfeld [1].

1. Classical Thermodynamics and Related Work

We propose a conceptual framework and a set of heuristics and methods for object randomisation based on computability and algorithmic probability different to those from classical methods that offer a refinement on the *Principle of Maximum Entropy* and also a different perspective of thermodynamics of computation from the point of view of algorithmic information.

Conventionally, and traditionally, thermodynamic entropy is defined as follows:

- A measure of *statistical* disorder
- Some quantity or property that increases but never decreases
- A process that defines the direction of time
- A measure of *statistical* information

Some of the problems surrounding the characterisation of the second law and entropy go back about a hundred years, to the time when they were introduced. While most of the discussion around thermodynamics is not only legitimate but central to the most pressing and important questions in physics, the statistical version of entropy has found a renewed application in the form of the so-called *Principle of Maximum Entropy*, often denoted by *Maxent*.

Previous work has considered the question of replacing all or part of the statistical machinery from statistical mechanics in order to arrive at an algorithmic approach to thermodynamics. Some authors have discussed the analogy between algorithms and entropy [2,3]. One example is the thought experiment "Maxwell's demon"—whereby Maxwell suggested that the second law of thermodynamics might hypothetically be violated by introducing intelligence in the form of a being capable of following an algorithm that enabled it to distinguish between particles of high and low energy—taken together with Szilard's discussion of this paradox [4]. Here, we offer an alternative framework in terms of which Maxwell's paradox and other thermodynamic-like phenomena can be reformulated in terms of computer programs via algorithmic probability and program-size complexity.

Other examples combining computation and entropy can be found in Seth Lloyd's concept of *thermodynamic depth* [5], heavily indebted to the work of Kolmogorov and Chaitin and to Bennett's notion of *logical depth* [6]; in Baez's *algorithmic thermodynamics* [7] approach that is capable of defining an algorithmic version of *algorithmic temperature* and *algorithmic pressure*, and in Crutchfield's *computational mechanics* using *epsilon-machines* [8]. Zurek has also proposed the inclusion of algorithmic randomness in a definition of physical entropy to allow the reformulation of aspects of thermodynamics, such as the Maxwell demon's thought experiment [9,10].

The interest in introducing algorithmic complexity to questions of thermodynamics and the second law derives from a wish to introduce an additional dimension into the discussion of the foundations of probability going back to Kolmogorov [11], Solomonoff [12], Chaitin [13] and Levin [14] among others, but also from an interest in the relationship between physics and logical information [3–7,9,10]. This includes testing or refining the second law of thermodynamics by taking into consideration and ruling out apparent disordered states that are not algorithmically random. Unlike statistical mechanical approaches, algorithmic complexity represents a generalisation over entropy that assigns lower entropy to objects that not only appear statistically simple but are also algorithmically simple by virtue of having a short generating mechanism capable of reproducing the algorithmic content of a system. Without such an additional dimension, uncomputable or algorithmically random and computable and non-algorithmically random networks are conflated and collapsed into the typical Bernoulli distribution produced by entropy, in which maximal entropy represents apparent statistical disorder, without distinguishing between networks with an algorithmic construction and algorithmic randomness. Indeed, a random-looking system with maximal entropy can be recursively generated by a short computer program that entropy would classify as statistically random but not as algorithmically random. An example of a fundamental limitation of Shannon entropy is, for example, offered in [15], where its fragility is exposed in a very simple example involving an attempt to quantify the deterministic vs. random nature of exactly the same object (a recursive graph of algorithmic randomness).

While we demonstrate that the mathematics of changes in algorithmic complexity and the formal repurposing capabilities of computer programs show a thermodynamic-like phenomenon that is similar, if not equivalent, to the mathematics of physical thermodynamics, in this paper we do not aim to connect physical thermodynamics to algorithmic thermodynamics directly. In other words, while we may be able to count the number of operations for converting one computer program into another by targeting a specific desired output from these programs, we do not enter in this paper into the details of the energetic cost of implementing these operations in hardware. We believe, however, that these results, while abstract in nature, are fundamentally "analogous" in character to the same kind of thermodynamic principles in classical mechanics. While this is beyond the scope of the present paper, we are motivated by the current results to revisit this important topic.

2. Notation and Definitions

Let G be a graph composed of a set $E(G)$ of edges with end points in $V(G)$ also known as the nodes of G. We will denote by $|V(G)|$ the cardinality of $V(G)$ and by $|E(G)|$ the cardinality of the set $E(G)$.

Definition 1. *The degree of a node in G is the number of directed and undirected edges linked to that node.*

Definition 2. *The degree sequence of a graph is the unsorted list of degree nodes of all nodes of G.*

Generally speaking, random graph is described by a probability distribution, or by a random process which generates them. Very intuitively, a random graph is obtained by starting with a set of number of isolated vertices and adding successive edges between them at random. Diverse models has been suggested to generate A random graph [16–18]. Each of these models produces different probability distributions on graphs. The most commonly studied model, Erdös–Rényi model, assigns equal probability to all graphs with equal number of edges, i.e., in Erdös–Rényi or E-R graph, each edge is included in the graph with probability p independent from every other edge.

The parameter $0 \geq p \leq 1$ determines the graph density and as it increases the model becomes more and more likely to include graphs with more edges. The value $p = 0.5$ corresponds to the case where all graphs on n vertices are chosen with equal probability and degree sequence approximating a uniform distribution, i.e., almost surely most nodes have about the same node degree. An E-R random graph can be obtained by starting with a set of n isolated nodes and adding random edges between them. Let ϕ be a property of G. We can quantify the probability of graph G to have that particular property $Prob(\phi(G))$ based on the set of elements from the distribution of the mathematical model used to produce the random graph. Generally, the probability of property ϕ can take values between 0 and 1 as $V(G)$ or $E(G)$ tends to infinity. These models can be used in the probabilistic method to provide a definition of what it means for a property to hold for almost all graphs. We can speak of the set of first-order properties of a graph for which the previous value is 1, rather than 0; let us refer to such properties as "holding almost surely". In the case of properties of E-R graphs, we always mean properties that "hold almost surely" and which can be satisfied by graphs on at most $|V(G)|$ vertices. We often shorten it to read "G has this property" meaning "sharing many of the well-known properties that hold almost surely for ER graphs" and we refer to Erdös–Rényi-like graphs as E-R graphs.

Definition 3. *An object α is algorithmic random if there exists a constant c such that for all n, $K(\alpha|n) \leq n - c$, where $\alpha|n$ denotes the first n bits of α. i.e., an algorithmic random object is almost algorithmically incompressible [13,14].*

Definition 4. *We define the elements of an object G that are able to "move" G towards algorithmic randomness (i.e., find the elements of the original graph that can produce another more or less algorithmic random graph) as $N(G)$ standing for the set of elements that can transform G' into a more random graph by at least $log_2(|G|)$ bits; and as $P(G)$ the set of elements that make G away from randomness.*

Definition 5. *The complementary set of $N(G) \cup P(G)$ is the set of neutral elements and are those that upon removal they do not change G neither towards greater or lower algorithmic randomness. If not otherwise specified, the neutral set will also contain the elements that "move" G towards or away from algorithmic randomness by less than a constant $log_2(n) + c$, with c a small positive real number [19,20].*

Definition 6. *We call the ordered set of all elements of G, σ_G, and their algorithmic contribution values sorted from most positive to most negative contribution to the algorithmic content of G and to $\sigma_N(G)$ and $\sigma_P(G)$, the negative and positive parts, i.e., the ordered sets of $N(G)$ and $P(G)$.*

We call $\sigma(G)$ the "signature" of G.

Definition 7. *We call a deletion an algorithmically random deletion (or simply a deletion if not otherwise stated), if it is non-recursive, non-computable or non-enumerable (which are used as synonyms and opposite of computable, algorithmic or recursive), that is, it cannot be algorithmically recovered or if its information was not recorded by, e.g., an index from an enumeration of elements of G when deleted, and is thus, for all computable purposes, lost.*

Definition 8. *An algorithmically random graph is a graph whose $P(G)$ is empty, that is, it has no elements that can move it away from randomness.*

Definition 9. *An algorithmically random Erdös–Rényi (E-R) graph is an E- R graph with $|P(G)| = 0$*

Not all E-R graphs are algorithmically random (see Theorem 1).

Graph Entropy

A major challenge in science is to provide proper and suitable representations of graphs and networks for use in fields ranging from social sciences to physics [21] and chemistry [22].

Networks have been characterised using classical information theory. One problem in this area is the interdependence of many graph-theoretic properties, which makes measures more sophisticated than single-property measurements [23] difficult to come by. The standard way to address this is to generate graphs that have a certain specific property while being random in all other respects, in order to check whether or not the property in question is typical among an ensemble of graphs with otherwise seemingly different properties. The Shannon entropy (or simply entropy) of a graph G can be defined by

$$H_i(G) = -\sum_{i=1}^{n} P(G(x_i)) \log_2 P(G(x_i)) \tag{1}$$

where G is the random variable with property i and n possible outcomes (all possible adjacency matrices of size $|V(G)|$). For example, a completely disconnected graph G with all adjacency matrix entries equal to zero has entropy $H(G) = 0$, because the number of different symbols in the adjacency matrix is 1. However, if a different number of 1s and 0s occur in G, then $H(G) \neq 0$.

Just as for strings, Shannon entropy can also be applied to the node degree sequence of a graph [24]. The Shannon entropy of an unlabelled network characterised by its degree distribution can be described by the same formula for Shannon entropy where the random variable is a degree distribution. The chief advantage of so doing is that it is invariant to relabellings. This also means that the degree distribution is not a lossless representation of a labelled network (but rather of its isomorphic group), and is an interesting entropic measure, but one that can only be used when the node labels are not relevant. For example, in a causal recursive network, the node labels may represent time events in a sequence that has a meaning captured in the network labelling, in which case the degree distribution sequence (where no agreement has been reached on the order of the elements in the distribution sequence,

which is therefore disordered) cannot be used to reconstruct the original network represented by the unlabelled version or the isomorphism group of the labelled networks. It is also clear that the concept of entropy rate cannot be applied to the degree distribution, because the node degree sequence has no particular order, or any order is meaningless because any label numbering will be arbitrary. This also means that Shannon entropy is not invariant to the language description of a network, especially as a labelled or an unlabelled network, except for clearly extreme cases (e.g., fully disconnected and completely connected networks, both of which have flat degree distributions and therefore the lowest Shannon entropy for degree sequence and adjacency matrix). While the application of Entropy to graph degree distributions has been relatively more common, the same Entropy has also been applied to other graph features, such as functions of their adjacency matrices [25], and to distance and Laplacian matrices [26]. A survey contrasting adjacency matrix based (walk) entropies and other entropies (e.g., on degree sequence) is offered in [25]. It finds that adjacency based ones are more robust vis-á-vis graph size and are correlated to graph algebraic properties, as these are also based on the adjacency matrix (e.g., graph spectrum). In general, we use Block entropy in order to detect more graph regularities (through the adjacency matrix) at a greater resolution. However, for Block entropy there is an extra factor to be taken into account. The unlabelled calculation of the Block entropy (not relevant for 1-bit entropy) of a graph has to take into consideration all possible adjacency matrix representations for all possible labellings. Therefore, the Block entropy of a graph is given by:

$$H(G) = \min\{H(A(G_L))|G_L \in L(G)\} \tag{2}$$

where $L(G)$ is the group of all possible labellings of G.

In [15], we introduced a graph that is generated recursively by a small computer program of (small) fixed length, yet when looking at its degree sequence it tends to maximal entropy and when looking at the adjacency matrix it tends to zero entropy at the limit, thus displaying divergent values for the same object when considering different mass probability distributions—when assuming the uniform distribution to characterise an underlying ensemble comprising all possible adjacency matrices of increasing size, or when assuming all possible degree sequences in the face of a total lack of knowledge of the deterministic nature of the graph in question.

3. Algorithmic Information Dynamics

The expected value of algorithmic entropy equals its Shannon entropy up to a constant that depends only on the distribution [27]. That is, every deterministic source has both low entropy and low algorithmic randomness, and algorithmically random objects surely have the highest Shannon entropy. However, in practical terms, they are fundamentally different. Nowhere in Shannon entropy is there any indication as to how to estimate the underlying mass probability distribution needed to determine the random or deterministic nature of a source, the availability of some other method for doing so being simply assumed.

Algorithmic complexity, however, does provide many methods, albeit very difficult ones, to estimate the algorithmic randomness of an object by inspecting the set of possible programs whose size is at most the size of the shortest program that may produce the object. One popular way to approximate it has been by using lossless compression algorithms, given that a compressed program is sufficient proof of non-randomness.

The algorithmic (Kolmogorov–Chaitin) complexity of a string x is the length of the shortest effective description of x. There are several versions of this notion. Here, we use mainly the plain complexity, denoted by $C(x)$.

We work over the binary alphabet $\{0,1\}$. A string is an element of $\{0,1\}^*$. If x is a string, $|x|$ denotes its length. Let M be a universal Turing machine that takes two input strings and outputs one string. For any strings x and y, we define the algorithmic complexity of x conditioned by y with respect to M, as:

$$C_M(x|y) = \min\{|p| \text{ such that } M(p, y) = x\}.$$

We match the machine M with a universal machine U, thereby allowing us to drop the subscript. We then write $C(x|y)$ instead of $C_M(x|y)$. We also write $C(x)$ instead of $C(x|\lambda)$ (where λ is the empty string).

3.1. Approximations to Algorithmic Complexity

We have introduced methods to approximate the algorithmic complexity of a graph with interesting results [28–30]. For example, in [29], correlations were reported among algebraic and topological properties of synthetic and biological networks by means of algorithmic complexity, and an application to classify networks by type was developed in [30]. Together with [29,30], the methods introduced represent a novel view and constitute a formal approach to graph complexity, while providing a new set of tools for the analysis of the local and global structures of networks.

The algorithmic probability [12,14,31] of a string s, denoted by $AP(s)$ provides the probability that a valid random program p written in bits uniformly distributed produces the string s when run on a universal (prefix-free (the group of valid programs forms a prefix-free set (no element is a prefix of any other, a property necessary to keep $0 < AP(s) < 1$); for details, see [32,33])) Turing machine U. Formally,

$$AP(s) = \sum_{p:U(p)=s} 1/2^{|p|} \tag{3}$$

That is, the sum over all the programs p for which a universal Turing machine U outputs s and halts.

Algorithmic probability and algorithmic complexity K are formally (inversely) related by the so-called algorithmic Coding theorem [32,33]:

$$|-\log_2 AP(s) - K(s)| < \mathcal{O}(1) \tag{4}$$

As shown in [29], estimations of algorithmic complexity are able to distinguish complex from random networks (of the same size, or growing asymptotically), which are both in turn distinguished from regular graphs (also of the same size). K calculated by the BDM assigns low algorithmic complexity to regular graphs, medium complexity to complex networks following Watts–Strogatz or Barabási–Albert algorithms, and higher algorithmic complexity to random networks. That random graphs are the most algorithmically complex is clear from a theoretical point of view: nearly all long binary strings are algorithmically random, and so nearly all random unlabelled graphs are algorithmically random [34], where algorithmic complexity is used to give a proof of the number of unlabelled graphs as a function of its randomness deficiency (how far it is from the maximum value of $K(G)$).

The *Coding Theorem Method* (CTM) [35,36] is rooted in the relation specified by algorithmic probability between frequency of production of a string from a random program and its algorithmic complexity (Equation (4). It is also called the algorithmic *Coding theorem*, to contrast it with another coding theorem in classical information theory). Essentially, it uses the fact that the more frequent is a string (or object), the lower is its algorithmic complexity; and strings of lower frequency have higher algorithmic complexity.

3.2. Block Decomposition Method

The *Block Decomposition Method* (BDM) was introduced in [29,37]. It requires the partition of the adjacency matrix of a graph into smaller matrices using, which we can numerically calculate its algorithmic probability by running a large set of small two-dimensional deterministic Turing machines, and then—by applying the algorithmic Coding theorem—its algorithmic complexity. Then, the overall complexity of the original adjacency matrix is the sum of the complexity of its parts, albeit with a

logarithmic penalisation for repetition, given that n repetitions of the same object only add $\log n$ to its overall complexity, as one can simply describe a repetition in terms of the multiplicity of the first occurrence. More formally, the algorithmic complexity of a labelled graph G by means of BDM is defined as follows:

$$BDM(G,d) = \sum_{(r_u,n_u)\in A(G)_{d\times d}} \log_2(n_u) + K_m(r_u) \tag{5}$$

where $K_m(r_u)$ is the approximation of the algorithmic complexity of the subarrays r_u arrived at by using the algorithmic Coding theorem (Equation (4)), while $A(G)_{d\times d}$ represents the set with elements (r_u, n_u), obtained by decomposing the adjacency matrix of G into non-overlapping squares of size d by d. In each (r_u, n_u) pair, r_u is one such square and n_u its multiplicity (number of occurrences). From now on, $BDM(g, d = 4)$ is denoted only by $K(G)$, but it should be taken as an approximation to $K(G)$ unless otherwise stated (e.g., when taking the theoretical true $K(G)$ value). Once CTM is calculated, BDM can be implemented as a look-up table, and hence runs efficiently in linear time for non-overlapping fixed size submatrices.

3.3. Normalised BDM

A normalised version of BDM is useful for applications in which a maximal value of complexity is known or desired for comparison purposes. The chief advantage of a normalised measure is that it enables a comparison among objects of different sizes, without allowing size to dominate the measure. This is useful in comparing arrays and objects of different sizes. First, for a square array of size $n \times n$, we define:

$$MinBDM(n)_{d\times d} = \lfloor n/d \rfloor + \min_{x\in M_d(\{0,1\})} CTM(x) \tag{6}$$

where $M_d(\{0,1\})$ is the set of binary matrices of size $d \times d$. For any n, $MinBDM(n)_{d\times d}$ returns the minimum value of Equation (5) for square matrices of size n, so it is the minimum BDM value for matrices with n nodes. It corresponds to an adjacency matrix composed of repetitions of the least complex $d \times d$ square. It is the all-1 or all-0 entries matrix, because $0_{d,d}$ and $1_{d,d}$ are the least complex square base matrices (hence the most compressible) of size d.

Secondly, for the maximum complexity, Equation (5) returns the highest value when the result of dividing the adjacency matrix into the $d \times d$ base matrices contains the highest possible number of different matrices (to increase the sum of the right terms in Equation (5)) and the repetitions (if necessary) are homogeneously distributed along those squares (to increase the sum of the left terms in Equation (5)), which should be the most complex ones in $M_d(\{0,1\})$. For $n, d \in \mathbb{N}$, we define a function

$$f_{n,d} : M_d(\{0,1\}) \longmapsto \mathbb{N}$$

that verifies:

$$\sum_{r\in M_d(\{0,1\})} f_{n,d}(r) = \lfloor n/d \rfloor^2 \tag{7}$$

$$\max_{r\in M_d(\{0,1\})} f_{n,d}(r) \leq \; 1 + \min_{r\in M_d(\{0,1\})} f_{n,d}(r) \tag{8}$$

$$CTM(r_i) > CTM(r_j) \;\Rightarrow\; f_{n,d}(r_i) \geq f_{n,d}(r_j) \tag{9}$$

The value $f_{n,d}(r)$ indicates the number of occurrences of $r \in M_d(\{0,1\})$ in the decomposition into $d \times d$ squares of the most complex square array of size $n \times n$. The condition in Equation (7) establishes that the total number of component squares is $\lfloor n/d \rfloor^2$. The condition in Equation (8) reduces the square repetitions as much as possible, so as to increase the number of differently composed squares

as far as possible and distribute them homogeneously. Finally, Equation (9) ensures that the most complex squares are the best represented. Then, we define:

$$MaxBDM(n)_{d\times d} = \sum_{\substack{r \in M_d(\{0,1\}),\\ f_{n,d}(r) > 0}} \log_2(f_{n,d}(r)) + CTM(r)$$

Finally, the normalised BDM value of an array X is:
Given a square matrix X of size n, $NBDM(X)_d$ is defined as

$$\frac{CTM(X) - MinBDM(n)_{d\times d}}{MaxBDM(n)_{d\times d} - MinBDM(n)_{d\times d}} \tag{10}$$

This way we take the complexity of an array X to have a normalised value which is not dependent on the size of X but rather on the relative complexity of X with respect to other arrays of the same size. The use of $MinBDM(n)_{d\times d}$ in the normalisation is relevant. Note that the growth of $MinBDM(n)_{d\times d}$ is linear with n, and the growth of $MaxBDM(n)_{d\times d}$ exponential. This means that for high complexity matrices, the result of normalising by using just $CTM(X)/MaxBDM(n)_{d\times d}$ would be similar to $NBDM(X)_d$. However, it would not work for low complexity arrays, as, when the complexity of X is close to the minimum, the value of $CTM(X)/MaxBDM(n)_{d\times d}$ drops exponentially with n. For example, the normalised complexity of an empty array (all 0s) would drop exponentially in size. To avoid this, Equation (10) considers not only the maximum but also the minimum.

Notice the heuristic character of $f_{n,d}$. It is designed to ensure a quick computation of $MaxBDM(n)_{d\times d}$, and the distribution of complexities of squares of size $d \in \{3,4\}$ in $D(5,2)$ ensures that $MaxBDM(n)_{d\times d}$ is actually the maximum complexity of a square matrix of size n, but for other distributions it could work in a different way. For example, the condition in Equation (8) assumes that the complexities of the elements in $M_d(\{0,1\})$ are similar. This is the case for $d \in \{3,4\}$ in $D(5,2)$, but it may not be true for other distributions. However, at any rate, it offers a way of comparing the complexities of different arrays independent of their size.

3.4. A Reprogrammability Calculus

At the core of the (re-)programmability analysis is a causal calculus introduced in [38] based on Algorithmic Information Dynamics, the change in complexity of a system subject to perturbations, particularly the direction (sign) and magnitude of the change in algorithmic information content C between wild and perturbed instants of the same object, such as objects G and G', which for purposes of illustration can be graphs with a set of nodes $V(G)$ and a set of edges $E(G)$.

We define two measures of reprogrammability (denoted by $P_R(G)$ and $P_A(G)$ as introduced in [38]) as measures that capture different aspects of the capability of a system's elements to move an object G towards or away from randomness.

Definition 10. *Relative programmability is defined as* $P_R(G) := MAD(\sigma(G)))/n$ *or 0 if* $n = 0$, *where* $n = \max\{|\sigma(G)|\}$ *measures the shape of* $\sigma_P(G)$ *and how it deviates from other distributions (e.g., uniform or normal), and MAD is the median absolute deviation of G.*

Definition 11. *Absolute programmability is defined as* $P_A(G) := |S(\sigma_P(G)) - S(\sigma_N(G))|/m$, *where* $m = \max(S(\sigma_P(G)), S(\sigma_N(G)))$ *and S is an interpolation function. This measure of programmability captures not only the shape of* $\sigma_P(G)$ *but also the sign of* $\sigma_P(G)$ *above and below* $x = 0$.

The dynamics of a graph can then be defined as transitions between different states, and one can always ask after the potential causal relationship between G and G'. In other words, what possible underlying minimal computer program can explain G' as evolving over discrete time from state G?

For graphs, we can allow the operation of edge e removal from G denoted by $G\backslash e$ where the difference $|C(G) - C(G\backslash e)|$ is an estimation of the (non-)shared algorithmic mutual information of G and $G\backslash e$ or the *algorithmic information dynamics* (or *algorithmic dynamics* in short) for evolving time-dependent systems (e.g., if G' evolves from G after t steps). If e does not contribute to the description of G, then $|C(G) - C(G\backslash e)| \sim \log_2 |V(G)|$, where $|V(G)|$ is the node count of G, i.e., the algorithmic dynamic difference will be very small and at most a function of the graph size, and thus the relationship between G and G' can be said to be causal and not random, as G' can be derived from G' with at most $\log_2 |V(G)|$ bits. If, however, $|C(G) - C(G\backslash e)| > \log_2 |V(G)|$ bits, then G and $G\backslash e$ are not states having the same causal origin and the algorithmically random removal of e results in a loss because $G\backslash e$ cannot recursively/computably recover G and has to be explained independently, e.g., as noise. In contrast, if $C(G) - C(G\backslash e) < \log_2 |V(G)|$, then e can be explained by G alone and it is algorithmically not contained/derived from G, and is therefore in the causal trajectory of the description of G from $G\backslash e$ and of $G\backslash e$ from G.

If G is random, then the effect of e will be small in either case, but if G is richly causal and has a very small generating program, then e as noise will have a greater impact on G than would removing e from the description of an already short description of G. However, if $|C(G) - C(G\backslash e)| \leq \log_2 |V(G)|$, where $|V(G)|$ is the vertex count of G, then e is contained in the algorithmic description of G and can be recovered from G itself (e.g., by running the program from a previous step until it produces G with e from $G\backslash e$).

4. The Thermodynamics of Computer Programs

A thermodynamic-like result is illustrated by a measure of *sophistication* based on quantifying the difficulty of reprogramming an object according to its underlying algorithmic generator or computer program. A measure of sophistication is a measure capable of telling apart "sophisticated" cases from simple and random objects, the latter being assigned low complexity, as in the case of Bennett's logical depth [6].

For example, in a complete graph G, the algorithmic-random deletion of any single node or single edge has the same effect, because all nodes and all edges make the same algorithmic-information contribution to the original graph as they cannot be distinguished in any way, and so their recovery does not need recording in, e.g., an enumerable index. In this case, for a complete graph, the ordered elements $\sigma(G)$ can be analytically derived from the uniform mass probability distribution defined by $x = \log_2 |V(G)|$ with $|V(G)|$ the node count of G because all nodes in a complete graph are neutral since they contribute by $\log_2 |V(G)|$. In contrast, the algorithmically random deletion of any single edge does not lead to a complete graph, and after two edge deletions, recovering any single one is not algorithmic, as it entails a lack of recording. All edges are therefore in $N(G)$ as they move the complete graph G towards algorithmic randomness.

Figure 1 can help illustrate how uniform random perturbations may provide a picture of the set of possible future states and may be relevant in the case of naturally evolving systems. This means that, in both cases in Figure 1, the asymmetric cost of moving random to simple and simple to random from a purely algorithmic perspective may also be relevant in the case of naturally evolving systems. The degree to which the perturbations are informative depends on whether the dynamic evolution process is proceeding (nearly) uniformly at random or not. An alternative way to illustrate the phenomenon of asymmetric reprogrammability is by considering networks as produced by computer programs (Figure 2) (or as produced by computer programs). The algorithmic complexity C of a complete graph k grows by its number of nodes because the generating mechanism is of the form "connect all N nodes", where $N = |V(k)|$. In contrast, the algorithmic complexity of an algorithmically random Erdös–Rényi (E-R) graph with edge density $\sim$0.5 grows by the number of edges $|E(\text{E-R})|$ because to reproduce a random graph from scratch the sender would need to specify every edge connection, as there is no way to compress the description.

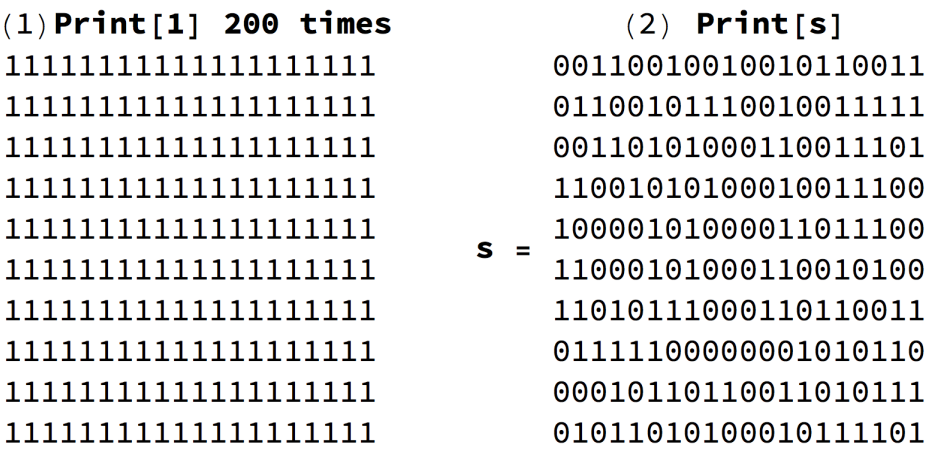

Figure 1. The thermodynamics of computer programming. (**Top**) The programs producing simple versus random data have different reprogrammability properties. If repurposed to generate programs to print blocks of 0s, we only need a single intervention in the generative program of (**1**), changing 1 to 0 inside the Print instruction indicating that 200 0s be printed instead of 200 1s. In contrast, asking a program that prints a random binary string s to print only 0s will require on average $|s|/2$ interventions to manually change every bit 1 to 0. Random perturbations can be seen as the exploration of the possible paths through which an object may evolve over time.

As depicted in Figure 2, algorithmic-random removal of a node n from a complete graph k produces another complete, albeit smaller graph. Thus, the generating program for both k and $k' = k \backslash n$ is also the relationship between k and k', which is of a causal nature. In contrast, if an edge e is removed from k, the generating program of $k'' = k \backslash e$ requires the specification of e and the resulting generating program of size $C(k'') > C(k)$, sending k towards randomness for every edge e algorithmic-randomly removed.

On the one hand, moving a complete graph toward randomness (see Figures 1 and 2) requires algorithmically random changes. On the other hand, we see how a random graph can also be easily rewired to be a complete graph by edge addition. However, if the complete graph is required to exactly reproduce a specific random graph and not just its statistical properties, then one would need to have full knowledge of the specific random graph and apply specific changes to the complete graph, making the process slow and requiring additional information. However, moving the random graph toward a complete graph requires the same effort as before, because the complete graph is unique, given the fixed number of nodes and implies reaching edge density 1 no matter the order of the added edges. Nevertheless, transforming a simple graph, such as the complete graph (see Figure 2), by algorithmically random one-by-one edge removal has a greater impact on its original algorithmic complexity than performing the same operation on a random graph.

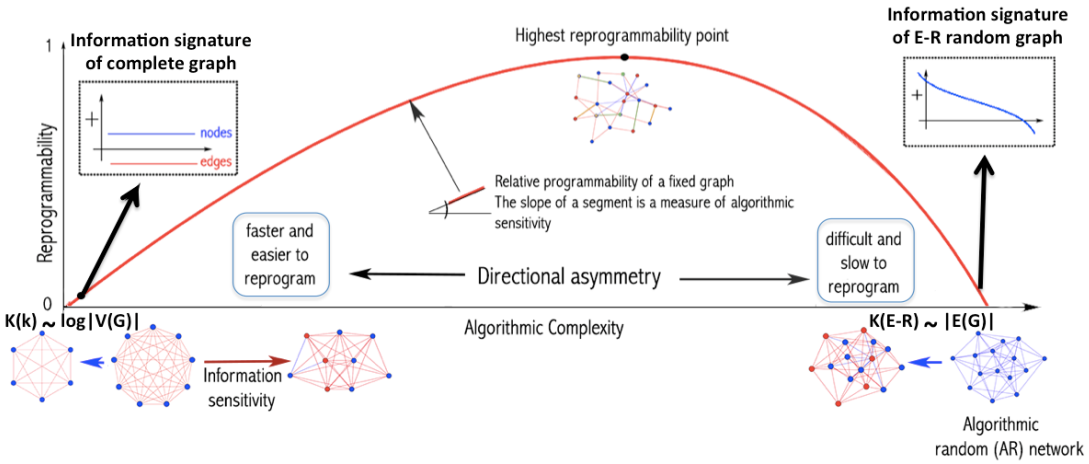

Figure 2. Graphs as produced by programs. All real-world networks lie between the extreme cases of being as simple as a complete graph whose algorithmic complexity C is minimal and grows by only $\log|V(k)|$, and a random (also statistically random and thus E-R) graph whose algorithmic complexity is maximal and grows by its number of edges $|E(\text{E-R})|$. If we ask what it takes to change the program producing k to produce E-R and vice versa, in a random graph, any single algorithmic-random node or edge removal does not entail a major change in the program-size of its generating program, which is similar in size to the random graph itself, i.e., $|E(G)|$. The curve shows how, without loss of generality, the reprogramming capability of networks, as produced by computer programs, produces an asymmetry imposed by algorithmic complexity and reminiscent of traditional thermodynamics as based on classical probability. A maximally random network has only positive (blue) elements (Figures 3 and 4) because there exists no perturbation that can increase the randomness of the network either by removing a node or an edge, as it is already random (and thus non-deterministic). Thus, changing its (near) minimal program-size length by edge or node removal is slow. However, a simple graph may have elements that incline its program-size length toward randomness. In each extreme case (simple vs. random), the distribution of sorted elements capable of shifting in each direction is shown in the form of what we call "signatures", both for algorithmically random edge and node removal. The highest reprogrammability point is the place where a graph has as many elements to steer it in one direction as in the other.

Graphs as Computer Programs

Specifically, if S is a simple graph and R an algorithmically random one, then we have it that $C(S) - C(S\backslash e) \geq C(R) - C(R\backslash e)$, i.e., the rate of change from S to (a non-specific) R is greater than in the other direction, thus imposing a thermodynamic-like asymmetry related to the difficulty of reprogramming one object into another according to its initial program-size complexity. This is because a random deletion to R will always have little impact on the underlying $print(R)$ with respect to $|R|$ (of at most logarithmic effect), because $print(R) \sim R$, but a random deletion (or a random transformation in general) to S may lead to a greater disruption of the small (by definition) generating program of S. The asymmetric axis where the highest reprogrammability point can be found is exactly the point at which $C(S) - C(S\backslash e) = C(R) - C(R\backslash e)$ for a specific S and R.

It is clear then how analysing the contribution of each element to the object, as shown in Figure 2, has the potential to reveal the algorithmic nature of the original object and how difficult it is to reprogram the underlying generative computer program in order to produce a different output/graph.

A thermodynamic-like effect can be found in the (re)programmability capabilities of an object. Moving random networks by edge removal is significantly more difficult than moving simple networks towards randomness. For random graphs, there are only a few elements, if any, that can be used to convert them slowly towards simplicity, as shown in Figure 2. In contrast, a larger number of elements can nudge a simple network faster towards randomness. This relationship, captured by the

reprogrammability rate for simple versus random graphs, induces a thermodynamic-like asymmetry based on algorithmic complexity and reprogrammability.

Figure 2 illustrates that, in a complete graph, the algorithmic-random removal of any single node leads to a less than logarithmic reduction in its algorithmic complexity, while the removal of any single edge leads to an increase in randomness. The former because the result is simply another complete graph of a smaller size, and the latter because an algorithmically random deleted link would need to be described after the description of the complete graph itself. In other words, if k_n = complete graph on n nodes, $C(k_n)$ is at most $C(n) + O(1) \leq log_2(n) + O(1)$ versus $C(k_{n-1}) \leq k(n-1) + O(1) \leq log_2(n-1) + O(1)$, thus the difference between $log_2(n)$ and $log_2(n-1)$ is very small and not even the difference between, e.g., n and $n - log_2(n)$, which is what one would normally call a "logarithmic additive difference".

It is worth noting that, if we only want to delete a single edge from the complete graph, there are $\binom{|V(k_n)|}{2}$ possibilities. However, there is only one such possibility up to isomorphism. Thus, if we only care about describing graphs up to isomorphism (or unlabelled), the complexity drops only by a constant. This is true for deleting any n edges, as there will then only be n possible isomorphisms. For this and the general case, we have previously investigated the relationship between labelled and unlabelled graph measures of algorithmic complexity [29,39,40], and, while there are some differences, for the most part the results hold, although they warrant some minor variations.

If a graph is evolving deterministically over time, its algorithmic complexity remains (almost) constant up to a logarithmic term as a function of time, because its generating mechanism is still the same as in the complete graph (which is a simplest extreme case), but if its evolution is non-deterministic and possible changes to its elements are assumed to be uniformly distributed, then a full perturbation analysis can simulate their next state, basically applying exhaustively all possible changes one step at a time. In this case, any node/edge perturbation of a simple graph has a very different effect than performing the same interventions on a random graph.

5. Principle of Maximum Algorithmic Randomness (MAR)

There is a wide range of applications in science, in particular in statistical mechanics, that serve as ways to study and model the typicality of systems and objects (see Section 6). Indeed, determining how removed an object is from maximum entropy has been believed to be an indication of its typicality and null model based on its information content [41].

Maximum entropy, or simply Maxent, is taken as the state of a system when it is at its most statistically disordered—when it is supposed to encode the least information.

We show here, however, that entropy may collapse cases that are only random-looking but are not truly so. Based on the ideas above, we can, nevertheless, suggest a conceptual and numerical refinement of classical Maxent by an algorithmic Maxent. Instead of making comparisons against a maximal entropy graph, one can make comparisons against a generated maximal algorithmic random graph (MAR) candidate (shown in red in Figure 3) by either comparing the original graph (denoted by G in Figure 3) or a compressed version (denoted by $minG$) approximated by, for example, algorithmic graph sparsification [42]. Such comparisons are marked as t and t' in Figure 3 and replace the need for comparison with a maximal entropy graph that may not be algorithmically random. Figure 3 shows how such a replacement can be made and what comparisons are now algorithmic (t and t') versus only statistical, which in the case of this illustration involves a graph that has been shown to produce a near maximal degree sequence when it is of lowest algorithmic randomness [15] (in magenta, top right).

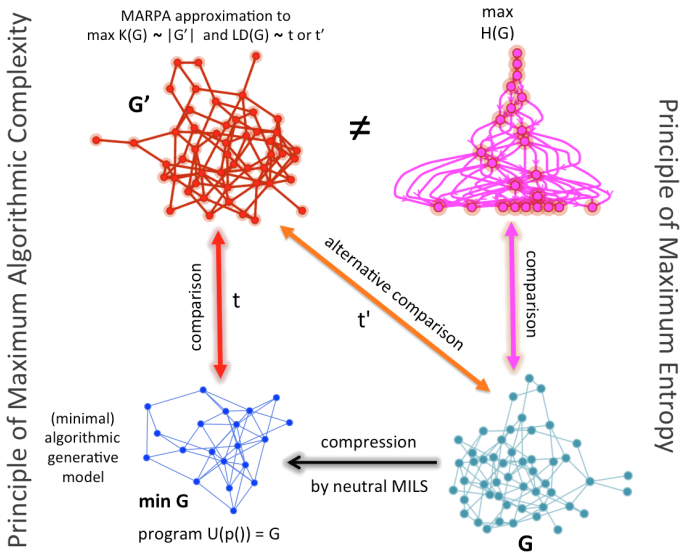

Figure 3. Algorithmic Maxent in the generation of graphs (without loss of generality, no other restrictions are imposed). The paths to statistical and algorithmic randomness are different, and they determine different principles for different purposes. Algorithmic Maxent tells apart recursive cases from non-recursive (thus algorithmic random) ones. Classical Maxent quantifies statistical randomness but its algorithmic refinement quantifies both statistical and algorithmic randomness. This opens up the range of possibilities for moving toward and reaching a random graph, by not only considering whether it is random-looking but also whether it is actually algorithmically random. Because the worse case for algorithmic Maxent is not to distinguish recursive cases (due to, e.g., semi-computability) its worse performance is to retrieve the same Gibbs measure and produce the same ensemble as classical Maxent would do.

5.1. Maximal Algorithmic Randomness Preferential Attachment (MARPA) Algorithm

Once the number of nodes is fixed, a MAR graph is of density 0.5, as is a classical E-R graph. This is because the highest algorithmic complexity is reached when $K(G) \sim |E(G)|$ is maximised somewhere between the two extreme cases in which fully disconnected and complete graphs reach minimum complexity $K(G) \sim log_2|V(G)|$.

This means that, without loss of generalisation to other objects, a Maximal Algorithmic Random graph G is an Erdös –Rényi (E-R) graph that is algorithmically random, i.e., whose shortest possible computer description is not (much) shorter than $|E(G)|$, where $|E(G)|$ is the number of edges of G; or, $|E(G)| - C(G) < c$.

MARPA seeks to maximise the information content of a graph G by adding new edges (or nodes) at every step. The process approximates a network of a given size that has the largest possible algorithmic randomness and is also an Erdös–Rényi (E-R) graph. An approximation of a "Maximal" Algorithmic-Random (MAR) graph can be produced as a reference object whose generating program is not smaller than the network (data) itself and can better serve in maximum (algorithmic-) entropy modelling.

MARPA allows constructions of a maximally random graph (or any object) by edge preferential attachment, in such a manner that randomness increases for any given graph. Let G be a network and $C(G\backslash e)$ the information value of e with respect to G such that $C(G) - C(G\backslash e) = n$. Let $P = \{p_1, p_2, \ldots, p_n\}$ be the set of all possible perturbations. P is finite and bounded by $P < 2^{|E(G)|}$ where $E(G)$ is the set of all elements of G, e.g., all edges of a network G. We can approximate the set of perturbations e' in P such that $C(G) - C(G\backslash e') = n'$ with $n' < n$. As we iterate over all e in G and apply the perturbations that make $n' < n$, for all e, we go through all $2^{|E(G)|}$ possible perturbations (one can start with all $|E(G)|$ single perturbations only) maximising the complexity of $G' = max\{G|C(G) - C(G\backslash e) = \{$max among all p in P and $e \in G\}$ (which may not be unique and

can build an algorithmic probability distribution to compare with, one that is complementary to the distribution of maximal entropy graphs).

We denote the set of maximal entropy graphs as $\{MaxS(G)\}$ and the set of maximal algorithmic randomness graphs as $\{MaxK(G)\}$, while approximations to $\{MaxK(G)\}$ are denoted by $\{MaxC(G)\}$.

Approximating methods allow tighter bounds for MAR graphs beyond entropy approaches. This algorithmic version expands and improves over classical Maxent (Figure 4), which is based on classical information theory and Shannon entropy. While Figure 4 may be misleading because there is no guarantee of uniqueness of G' such that $K(G')$ is of maximal algorithmic randomness (neither in theory nor practice), under some restriction, under classical Maxent, the number of algorithmically random graphs is strictly smaller than the number of high entropy graphs for the same size and under any other constraint (e.g., same degree sequence or edge density) and thus can be seen as a reduction of the ensemble size and a refinement of classical Maxent.

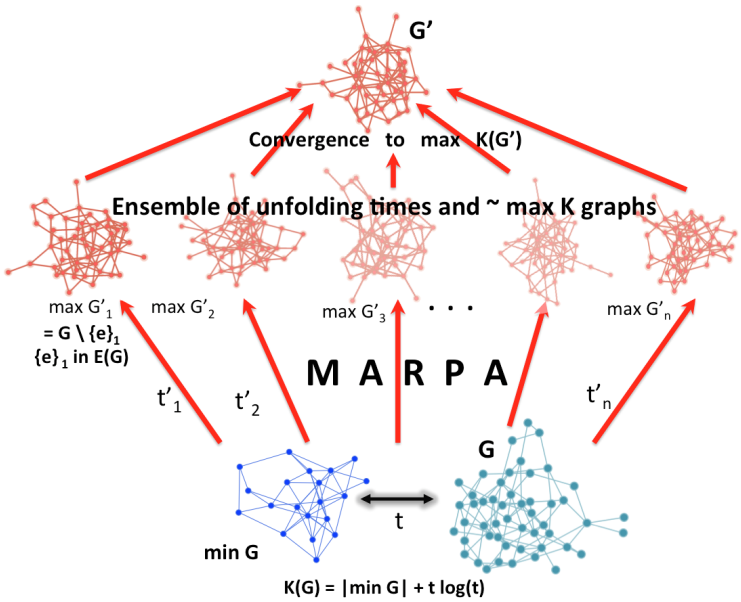

Figure 4. Reduction of the Gibbs/Boltzmann distribution. While objects may appear maximally statistically random in all other properties but the ones constrained by a problem of interest for which Maxent is relevant, there may actually be of different nature and some may actually be algorithmically random while others recursively generated hence they should not be placed in the same distribution. The original graph G has a shortest description $minG$ and perturbations to either the original and compressed version can lead to randomised graphs for different purposes. Some of them will have maximum entropy denoted by $maxG'_n$ but among them only some will also be of $maxK(G)$.

Definition 12. *The randomness deficiency of an object, such as a graph, in a particular distribution (e.g., a set of graphs) quantifies how far an object is from the greatest algorithmic random object with the same properties (e.g., edge density) in the distribution.*

The purpose of MARPA (Figure 4) is thus to maximise the randomness deficiency between an object of interest and its greatest algorithmic randomisation. To this end, it is necessary to estimate the algorithmic complexity of such an object by means such as popular lossless compression algorithms (e.g., Lempev-Ziv-Welch (LZW)), which are limited but are an improvement over other measures such as entropy [43], or by alternative means such as those introduced in [35–37,39,40] based on algorithmic probability.

Let such a constructed maximal complexity object $maxC(G)$ be used to quantify the randomness deficiency of an object of interest be denoted by $C(G)$. The procedure consists in finding the sequential set of perturbations $\{P\}$ that upon application to G leads to G' such that $maxC(G) - C(G')$ is

minimised, and where $C(G') - C(G)$ is the randomness deficiency estimation of G, i.e., how removed G is from its (algorithmic-)randomised version $maxC(G)$ (notice that $C(G)$ is upper bounded by $maxC(G)$ and the difference is always positive).

5.2. Supremacy of Algorithmic Maxent

To illustrate how algorithmic Maxent is an improvement over classical Maxent, we demonstrate that the E-R model admits different constructions, some of which are not algorithmic random.

Theorem 1. *(existence) At least one E-R graph exists which is not algorithmically random.*

Ackermann [44] and Rado [45] constructed the so-called Rado graph in a similar spirit to the ZK-graph [15] using an algorithmic/computable procedure based on what is called the BIT predicate by identifying the vertices of the graph with the natural numbers $0, 1, 2, \ldots$ An edge connects vertices x and y in the graph (where $x < y$) whenever the xth bit of the binary representation of y is nonzero. Thus, for example, the neighbours of vertex zero consist of all odd-numbered vertices, because the numbers whose 0th bit is nonzero are exactly the odd numbers. Vertex 1 has one smaller neighbour, vertex 0, as 1 is odd and vertex 0 is connected to all odd vertices. The larger neighbours of vertex 1 are all vertices with numbers congruent to 2 or 3 modulo 4, because those are exactly the numbers with a nonzero bit at index 1. The Rado graph is almost surely Erdös–Rényi on countably many vertices [44,45].

Theorem 2. *(non-uniqueness) There is more than one E-R graph that is not algorithmically random.*

The set of pseudo-random generating graphs with fixed seeds of different minimum program-size generates one E-R random graph, and for each different seed it will generate another E-R random graph.Therefore there more than one E-R graph that is not algorithmically random. it should be noted that an E-R graph with density 0.5 may be of maximal entropy, but can be produced by programs of varying length and thus of varying algorithmic randomness, i.e., either recursively generated or not. Figure 5 shows that the complexity of graphs with exactly the same number of nodes and edges that comply with the properties of an E-R graph (e.g., edge density ~ 0.5) do not comply with the property of maximum algorithmic randomness (MAR), and their values will diverge for typical randomly chosen examples of growing graphs by node count. It is also shown that the numerical approximations, both for simple (complete) and MAR graphs, follow the theoretical expectations. The proof is by induction.

5.3. Numerical Examples

Consider the absolute maximum algorithmic-random graph, which we denote by $maxC(G)$. $maxC(G)$ is a graph comprising the same number of nodes but with an edge rearrangement operation such that $C(G) < C(max(G)) \leq 2^k$, where $k = (|E(G)|(|E(G)| - 1))/4$ is the maximum number of edges in G divided by 2 where at edge density 0.5 it reaches maximal algorithmic randomness.

There are two possible methods to produce estimations of MAR graphs: one is top-down and one is bottom-up. The bottom-up method consists in adding edges starting from the empty graph until reaching the desired target network size (in nodes and/or edges).

The top-down method consists in deleting edges starting from a given network and finding the elements that maximise its algorithmic complexity. We call these methods edge-deletion and edge-addition algorithmic randomisations (as opposed to just randomisation in the classical sense). When constraining edge-addition by maximum number of nodes, both methods converge in maximal algorithmic value and statistical properties but may construct different graphs, both of which are estimation of MAR graphs.

A third method may consist in edge rearrangement/switching but this requires an exponentially greater number of operations at every step. A computationally more efficient way to produce an approximation to a MAR network by edge deletion is to start from a complete graph and keep deleting edges until a target network size is reached. The pseudo-code of the edge-deletion algorithmic randomisation is as follows:

1. Start with an empty or complete graph G.
2. Perform all possible perturbations for a single operation (e.g., edge deletion or addition) on G to produce G'.
3. Keep only the perturbation that maximised algorithmic randomness, i.e., $C(G') \geq C(G)$.
4. Set $G := G'$
5. Repeat 1 until final target size is reached or $C(G) > C(G')$.

With the cost function maximising algorithmic complexity, the above pseudo-code is valid for other operations to G' without loss of generalisation. These operations can be of other types, including edge rearrangement/switching and edge addition as well as edge removal (see Ref. [38]). However, different operations require different time complexities.

The edge-addition algorithmic randomisation can start from an empty set, for which one would be generating approximations of MAR graphs (for that operation) for all sizes from bottom up.

The algorithm can also be repeated to produce a set of candidate MAR graphs (not only because many MAR graphs may exist but because C is semi-computable and thus they are only approximations). The intersection of both $\{MaxC(G)\}$ and $\{MaxS(G)\}$ is an interesting candidate set with which to replace $\{MaxS(G)\}$ alone for purposes of Maxent. However, if the algorithm to produce $\{MaxC(G)\}$ approximates $\{MaxK(G)\}$ then $\{MaxS(G)\} \subset \{MaxC(G)\}$ and $\{MaxS(G)\}$ will be upper bounded by $\{MaxC(G)\}$ because it is upper bounded by $\{MaxK(G)\}$.

Figure 5 shows how MAR graph degree distributions follow E-R graphs for nodes with high degree but not for lower degree nodes. MAR graphs produce a degree sequence of greater entropy and algorithmic complexity because an algorithmic random graph should also have an algorithmic random degree sequence.

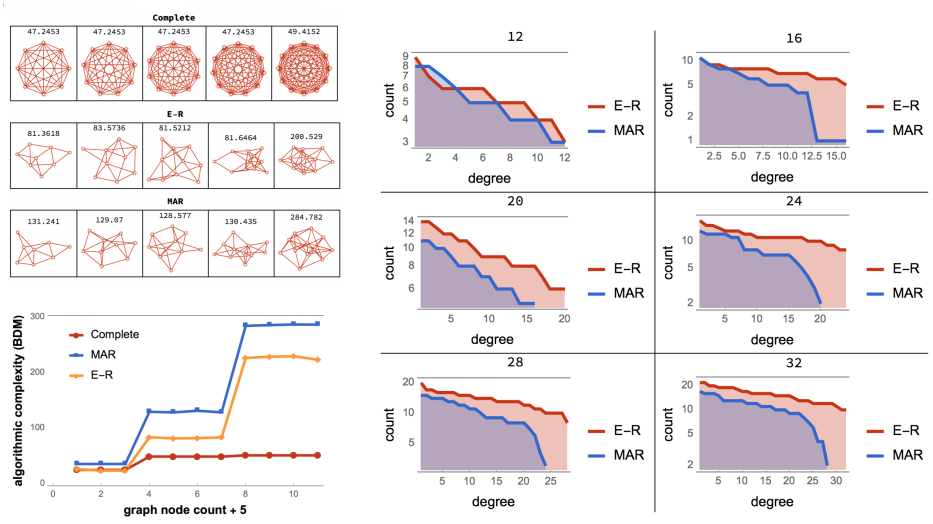

Figure 5. (**Top left**) One can produce a MAR graph starting from an empty graph and adding one edge at a time (see Figure 6) or one can start from a complete graph and start deleting edge by edge keeping only those that maximise the algorithmic randomness of the resulting graph. (**Bottom left**) Following this process, MAR graphs top E-R graphs meaning BDM effectively separate low algorithmic complexity (algorithmic randomness) from high entropy (statistical randomness), where entropy would simply be blind collapsing all recursive and non-recursive cases. (**Right**) degree distribution comparison between E-R and MAR graphs.

In Figure 6, the ensemble of E-R graphs are a randomisation of the MAR graphs with the same number of nodes. Despite allowing any possible number of edges, MAR graphs are equally random both according to classical and algorithmic (estimations) on their adjacency matrices but of greater entropy than E-R graphs. The ZK graph came high for degree sequence entropy as expected as it was designed for that purpose, but for approximations of algorithmic complexity by BDM, the ZK graph was considerably less random also as theoretically expected (as it was recursively generated [15]. MAR graphs, however, had both maximal entropy and maximal algorithmic randomness.

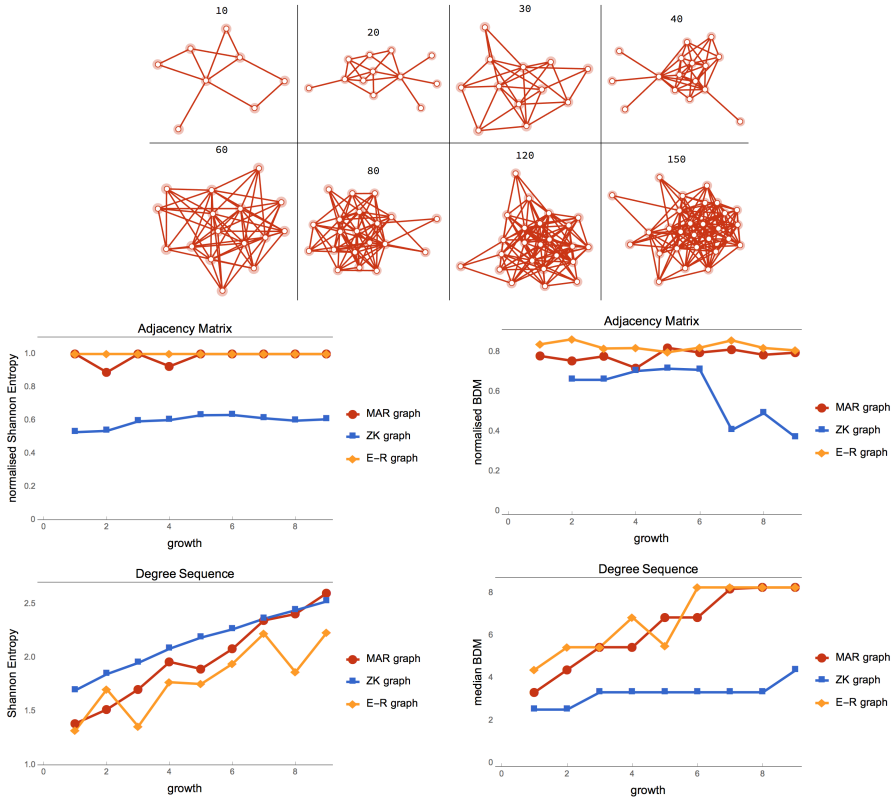

Figure 6. (**Top**) A MAR graph constructed by adding one by one every possible edge and keeping only those that maximise the uncompressibility/algorithmic complexity of the graph according to BDM. Shown are only eight steps from 10 edges to 150. (**Bottom**) MAR graphs versus ensemble of pseudo-randomly generated E-R graphs (with edge density 0.5) versus the ZK graph [15] designed to have minimal adjacency matrix entropy but maximal degree sequence entropy.

The edge density of the MAR graphs remains at 0.4 meaning that the greatest increase of algorithmic randomness is achieved by adding new nodes at about 0.4 density rather than increasing the density. When forcing to keep the same number of nodes and reach maximal algorithmic randomness, the algorithm does produce MAR graphs with 0.5 edge density.

Attempts to produce MAR graphs with popular lossless compression algorithms (Compress and Bzip2) failed because they were not sensitive enough to small changes required to evaluate the edge to add or delete to generate an approximation of a MAR graph.

Time Complexity and Terminating Criterion

Approximating a MAR graph candidate can be computationally very expensive, with exponential time complexity in the order $O(2^{n^2})$ for the bottom-up approach (adding edges) because, at every step, all possible additions have to be tested and evaluated against the original object. The time complexity is, however, reduced to linear time $O(n)$ for the top-down method (starting from a complete graph)

when only considering single-perturbation analysis, that is, instead of applying all possible subsets only individual ones are allowed.

Small MAR graphs can also be produced and are not necessary to recompute once the number of nodes (and possibly number of edges) is fixed. Better classical randomisations are also possible at reasonable time complexity, especially because they are only applied to graphs of fixed size and thus asymptotic time constraints are less relevant.

The terminating criterion for the top-down approach to produce MAR graphs consists in stopping the method either at the desired size by edge count or when further deletion leads to a decrease in complexity. In other words, starting from, e.g., a complete graph k, if $C(k) > C(k'^n)$ where n denotes the n iteration of the method description, then stop at iteration $n-1$ that maximises algorithmic randomness for all n. Otherwise, said $C(k) > C(k'^n)$ is concave as a function of n and the terminating time is at the global maximum of C.

6. Discussion

We have introduced a principle of maximum entropy (Maxent) based on computability, leading to a stronger principle based on algorithmic randomness, an approximation to a generalisation of the traditional *Maxent* based instead on algorithmic complexity approximations which are independent of the specific method such as the nature of the lossless compression algorithm.

Unlike E-R graphs, MAR graphs cannot be generated by computer programs of much smaller size than the edge count of the networks themselves. The intuition behind the construction of a MAR graph is that the shortest computer program (measured in bits) that can produce the adjacency matrix of a MAR graph, is of about the size of the adjacency matrix and not significantly shorter.

We believe that deconvolving aspects of randomness versus pseudo-randomness in scientific practice can help elucidate many confounding aspects within certain areas such as physics. For example, the so-called *Holographic Principle*, where it is often suggested that "all the information about the universe is contained in its surface", but makes no distinction between statistical and algorithmic information.

Roughly, if the universe has positive entropy, then the set of universes with the same probability distribution is much larger than the number of actual universes in which we live. For example, for a universe statistically random there are many more statistical-equivalent candidates, including those algorithmically generated reproducing the same statistical distribution (for an example of how an object can mimic to have a target probability distribution, appear random but not being so, see [15]), than candidate universes of low entropy universe. This is because for only for low entropy candidate universes will be also of low algorithmic randomness producing a small set of possible models, but for higher entropy cases, the number of high entropy but low algorithmic complexity universes diverge. In other words, there are about Q^{n-c} ways to describe Q particles contained in a universe of length n with less than c bits, thus $Q/S_Q!$ many ways to place Q particles in the same space, preserving the same statistical distribution (even under all coarse-graining partitions S_Q). S_Q is the subset of particles Q whose permutation in the same subset preserve entropy S. This is because we know that the number of objects in a set with highest algorithmic complexity is a proper subset of the number of objects with highest entropy in the same set. Theorems 1 and 2 are here relevant as they show this for a specific type of object, showing that there are E-R graphs that have the same properties but can be differently generated by recursive means. In other words, only a fraction of those would actually describe the exact configuration of our universe or some algorithmic equivalent one. The difference is huge considering that the particles that make a human being can be arranged in so many configurations other than a human being (let alone a particular one) preserving the same statistical properties of their original arrangement in a human being. However, it is highly algorithmically probable to define a configuration closer to a human being (because it is of low algorithmic complexity, and according to the coding theorem is algorithmically highly probable [33]), or even a particular one, when describing the process that led to that human being rather than simply describing its statistical properties.

The point is thus that information claims about statistical properties of objects is way less precise than what they apparently convey which is not enough to reconstruct the same object, such as our universe, or its particular generating code. A clear distinction should thus be made between statistical versus algorithmic information and statistical versus algorithmic randomness.

This algorithmic approach provides such a platform, independent of whether we can numerically or methodology achieve such distinction, we claim we can approximate. Our framework offers an alternative to translate and reinterpret generic examples from classical thermodynamics in the language of computer programs. For example, Maxwell's demon is captured or reformulated by a recursively enumerating process keeping record of algorithmic-random deletions, allowing reversing these deletions into non-algorithmic random ones and incurring in an unavoidable cost. The undeletion process then becomes computable by exchanging information loss with memory (e.g., a reverse lookup table), assigning an index to each process from a preconceived, well-defined enumerating scheme, making the deletion process reversible without algorithmic information loss. The incurred cost is in the recording of the data and the preconception of the computable enumeration from which the knowledge is extracted and recorded at every stage back and forth, as their answers can then be seen as coming from an Oracle machine (the algorithmic Maxwell demon) providing the answer to the question of which element is deleted at any given time to make a network more algorithmically random at no cost ("hotter"). The enumeration would then allow us to trace back and forth the location of every element independent of its cause and effect, breaking the asymmetry and apparently violating the asymmetry found in the otherwise natural thermodynamics of computer programs, as described here.

In the present work, we have only considered deletion (of edges or nodes) from a graph but this is without loss of generalisation. Indeed, all the results hold when considering other recursive vs. non-algorithmic transformations, such as swapping nodes or edges or swapping bits, rows and columns on its adjacency matrix. The algorithmic asymmetry in reprogrammability will come from whether such a specific transformation is computable (recursive) or algorithmically random. Indeed, the algorithmic information content properties of G are essentially identical to those of $t(G)$ if $G = t'(t(G))$, i.e., if the transformation t is enumerable and reversible but not if t is non-computable or algorithmically random. Edge deletion is, however, the most interesting approach in practice for cases such as sparse graphs, graph dimension reduction and network reconstruction.

Evidently, our results, methods, and Maxent refining algorithm will rely entirely on the numerical approximations to $K(G)$, and thus on how good the estimation $C(G)$ is. We have introduced in [35,36] novel methods that have yielded better estimations than those methods based on popular lossless compression algorithms [37,43] and on this basis we have discovered what we believe to be interesting applications [20,46,47]. For example, the use of popular lossless compression algorithms will make $\{maxS\} \cap \{maxC\} = \{maxS\} \cup \{maxC\}$ for most cases, where C is, for example, LZW, because LZW is closer to entropy S than to algorithmic complexity K [43,48]. A review of alternative measures for both entropy and algorithmic complexity is available in [49].

7. Conclusions

From an asymmetric relationship in a measure of reprogrammability, we conclude that the probabilistic basis of the second law of thermodynamics has strong similarities and connections to the difficulty of formally repurposing certain objects to induce them to behave in different computational ways.

We have thus established interesting parallels between computer programming as based on the dynamics of computer programs being repurposed and related to thermodynamic-like phenomena in terms of algorithmic probability, and algorithmic complexity, from which reprogrammability indexes can be defined. The lack of symmetry in the operation of rewiring networks with regards to algorithmic randomness implies a direction or an asymmetry, which indicates a natural direction for digital thermodynamics. After showing that the set of maximum entropy graphs is not the same as the set of maximum algorithmically complex graphs, we introduced a principle akin to maximum entropy but based on algorithmic complexity. The principle of maximum algorithmic randomness (MAR) can

therefore be viewed as a refinement of Maxent, as it is a proper subset of algorithmically random graphs, having also the highest entropy.

We believe that only by rigorously selecting models from data can we hope to make progress in understanding first principles and that current statistical approaches leave all the final interpretation to the observer and help little or only indirectly in truly exhibiting the properties that such approaches are supposed to grasp or reveal.

Author Contributions: Conceptualization, H.Z., N.A.K. and J.T.; Data curation, H.Z.; Investigation, H.Z.; Methodology, H.Z.; Software, H.Z.; Validation, H.Z.; Visualization, H.Z.; Writing—original draft, H.Z.; Writing—review & editing, H.Z., N.A.K. and J.T.

Funding: H.Z. wishes to received funding from the Swedish Research Council (Vetenskapsrådet) grant No. 2015-05299. J.T. received funding from the King Abdullah University of Science and Technology.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Angrist, S.W.; Helper, L.G. Order and Chaos. In *Laws of Energy and Entropy*; Basic Books: New York, NY, USA, 1967; p. 215.
2. Hammer, D.; Romashchenko, A.; Shen, A.; Vereshchagin, N. Inequalities for Shannon entropies and Kolmogorov complexities. *J. Comput. Syst. Sci.* **2000**, *60*, 442–464. [CrossRef]
3. Grünwald, P.; Vitányi, P. Shannon Information and Kolmogorov Complexity. *arXiv* **2004**, arXiv:cs/0410002.
4. Szilard, L. On the decrease of entropy in a thermodynamic system by the intervention of intelligent beings. *Z. Phys.* **1929**, *53*, 840–856. [CrossRef]
5. Lloyd, S.; Pagels, H. Complexity as thermodynamic depth. *Ann. Phys.* **1988**, *188*, 186–213. [CrossRef]
6. Bennett, C.H. Logical Depth and Physical Complexity. In *The Universal Turing Machine: A Half-Century Survey*; Herken, R., Ed.; Oxford University Press: Oxford, UK, 1988; pp. 227–257.
7. Baez, J.C.; Stay, M. Algorithmic Thermodynamics, Computability of the Physical. *Math. Struct. Comput. Sci.* **2012**, *22*, 771–787. [CrossRef]
8. Crutchfield, J.P.; Shalizi, C.R. Thermodynamic depth of causal states: Objective complexity via minimal representations. *Phys. Rev. E* **1999**, *59*, 275–283. [CrossRef]
9. Zurek, W.H. Algorithmic randomness and physical entropy. *Phys. Rev. A* **1989**, *40*, 4731. [CrossRef]
10. Zurek, W.H. Thermodynamic cost of computation, algorithmic complexity, and the information metric. *Nature* **1989**, *341*, 119–124. [CrossRef]
11. Kolmogorov, A.N. Three approaches to the quantitative definition of information. *Probl. Inf. Transm.* **1965**, *1*, 1–7. [CrossRef]
12. Solomonoff, R.J. A formal theory of inductive inference: Parts 1 and 2. *Inf. Control* **1964**, *7*, 224–254. [CrossRef]
13. Chaitin, G.J. On the length of programs for computing finite binary sequences. *J. ACM* **1966**, *13*, 547–569. [CrossRef]
14. Levin, L.A. Laws of information conservation (non-growth) and aspects of the foundation of probability theory. *Probl. Inf. Transm.* **1974**, *10*, 206–210.
15. Zenil, H.; Kiani, N.A.; Tegnér, J. Low Algorithmic Complexity Entropy-deceiving Graphs. *Phys. Rev. E* **2017**, *96*, 012308. [CrossRef] [PubMed]
16. Erdös, P.; Rényi, A. On Random Graphs I. *Publ. Math. Debrecen* **1959**, *6*, 290–297.
17. Erdös, P.; Rényi, A. On the evolution of random graphs. *Bull. Inst. Int. Stat* **1961**, *38*, 343–347.
18. Gilbert, E.N. Random graphs. *Ann. Math. Stat.* **1959**, *30*, 1141–1144. [CrossRef]
19. Zenil, H.; Kiani, N.A.; Tegnér, J. Algorithmic Information Dynamics of Emergent, Persistent, and Colliding Particles in the Game of Life. In *From Parallel to Emergent Computing*; Adamatzky, A., Ed.; Taylor & Francis/CRC Press: Boca Raton, FL, USA, 2019; pp. 367–383.
20. Zenil, H.; Kiani, N.A.; Zea, A.; Tegnér, J. Causal Deconvolution by Algorithmic Generative Models. *Nat. Mach. Intell.* **2019**, *1*, 58–66. [CrossRef]
21. Boccaletti, S.; Bianconi, G.; Criado, R.; Del Genio, C.I.; Gómez-Gardenes, J.; Romance, M.; Sendiña-Nadal, I.; Wang, Z.; Zanin, M. The structure and dynamics of multilayer networks. *Phys. Rep.* **2014**, *544*, 1–122. [CrossRef]

22. Chen, Z.; Dehmer, M.; Emmert-Streib, F.; Shi, Y. Entropy bounds for dendrimers. *Appl. Math. Comput.* **2014**, *24*, 462–472. [CrossRef]
23. Orsini, C.; Dankulov, M.M.; Colomer-de-Simón, P.; Jamakovic, A.; Mahadevan, P.; Vahdat, A.; Bassler, K.E.; Toroczkai, Z.; Boguñá, M.; Caldarelli, G.; et al. Quantifying randomness in real networks. *Nat. Commun.* **2015**, *6*, 8627.
24. Korner, J.; Marton, K. Random access communication and graph entropy. *IEEE Trans. Inf. Theory* **1988**, *34*, 312–314. [CrossRef]
25. Estrada, E.; José, A.; Hatano, N. Walk entropies in graphs. *Linear Algebra Appl.* **2014**, *443*, 235–244. [CrossRef]
26. Dehmer, M.; Mowshowitz, A. A history of graph entropy measures. *Inf. Sci.* **2011**, *181*, 57–78. [CrossRef]
27. Teixeira, A.; Matos, T.A.; Souto, A.; Antunes, L. Entropy Measures vs. Kolmogorov Complexity. *Entropy* **2011**, *13*, 595–611.
28. Zenil, H.; Kiani, N.A; Tegnér, J. Quantifying loss of information in network-based dimensionality reduction techniques. *J. Complex Netw.* **2016**, *4*, 342–362. [CrossRef]
29. Zenil, H.; Soler-Toscano, F.; Dingle, K.; Louis, A. Graph Automorphisms and Topological Characterization of Complex Networks by Algorithmic Information Content. *Physica A* **2014**, *404*, 341–358. [CrossRef]
30. Zenil, H.; Kiani, N.A.; Tegnér, J. Algorithmic complexity of motifs clusters superfamilies of networks. In Proceedings of the IEEE International Conference on Bioinformatics and Biomedicine, Shanghai, China, 18–21 December 2013.
31. Kirchherr, W.W.; Li, M.; Vitányi, P.M.B. The miraculous universal distribution. *J. Math. Intell.* **1997**, *19*, 7–15. [CrossRef]
32. Calude, C.S. *Information and Randomness: An Algorithmic Perspective*, 2nd. ed.; EATCS Series; Springer: New York, NY, USA, 2010.
33. Cover, T.M.; Thomas, J.A. *Elements of Information Theory*, 2nd ed.; Wiley-Blackwell: New York, NY, USA, 2009.
34. Buhrman, H.; Li, M.; Tromp, J.; Vitányi, P. Kolmogorov Random Graphs and the Incompressibility Method. *SIAM J. Comput.* **1999**, *29*, 590–599. [CrossRef]
35. Delahaye, J.-P.; Zenil, H. Numerical Evaluation of the Complexity of Short Strings: A Glance Into the Innermost Structure of Algorithmic Randomness. *Appl. Math. Comput.* **2012**, *219*, 63–77. [CrossRef]
36. Soler-Toscano, F.; Zenil, H.; Delahaye, J.-P.; Gauvrit, N. Calculating Kolmogorov Complexity from the Frequency Output Distributions of Small Turing Machines. *PLoS ONE* **2014**, *9*, e96223. [CrossRef]
37. Zenil, H.; Soler-Toscano, F.; Kiani, N.A.; Hernández-Orozco, S.; Rueda-Toicen, A. A Decomposition Method for Global Evaluation of Shannon Entropy and Local Estimations of Algorithmic Complexity. *Entropy* **2018**, *20*, 605. [CrossRef]
38. Zenil, H.; Kiani, N.A.; Marabita, F.; Deng, Y.; Elias, S.; Schmidt, A.; Ball, G.; Tegnér, J. An Algorithmic Information Calculus for Causal Discovery and Reprogramming Systems. *bioRxiv* **2018**, 185637. [CrossRef]
39. Zenil, H.; Soler-Toscano, F.; Delahaye, J.-P.; Gauvrit, N. Two-Dimensional Kolmogorov Complexity and Validation of the Coding Theorem Method by Compressibility. *PeerJ Comput. Sci.* **2015**. [CrossRef]
40. Zenil, H.; Kiani, N.A.; Tegnér, J. Methods of Information Theory and Algorithmic Complexity for Network Biology. *Semin. Cell Dev. Biol.* **2016**, *51*, 32–43. [CrossRef] [PubMed]
41. Bianconi, G. The entropy of randomized network ensembles. *EPL (Europhys. Lett.)* **2007**, *81*, 28005. [CrossRef]
42. Zenil, H.; Kiani, N.A.; Tegnér, J. Minimal Algorithmic Information Loss Methods for Dimension Reduction, Feature Selection and Network Sparsification. *arXiv* **2018**, arXiv:1802.05843.
43. Zenil, H.; Badillo, L.; Hernández-Orozco, S.; Hernández-Quiroz, F. Coding-theorem Like Behaviour and Emergence of the Universal Distribution from Resource-bounded Algorithmic Probability. *Int. J. Parallel Emerg. Distrib. Syst.* **2018**. [CrossRef]
44. Ackermann, W. Die Widerspruchsfreiheit der allgemeinen Mengenlehre. *Math. Ann.* **1937**, *114*, 305–315. [CrossRef]
45. Rado, R. Universal graphs and universal functions. *Acta Arith.* **1964**, *9*, 331–340. [CrossRef]
46. Gauvrit, N.; Zenil, H.; Soler-Toscano, F.; Delahaye, J.-P.; Brugger, P. Human Behavioral Complexity Peaks at Age 25. *PLoS Comput. Biol.* **2017**, *13*, e1005408. [CrossRef]
47. Soler-Toscano, F.; Zenil, H. A Computable Measure of Algorithmic Probability by Finite Approximations with an Application to Integer Sequences. *Complexity* **2017**, *2017*. [CrossRef]

48. Zenil, H. Algorithmic Data Analytics, Small Data Matters and Correlation versus Causation. In *Berechenbarkeit der Welt? Philosophie und Wissenschaft im Zeitalter von Big Data (Computability of the World? Philosophy and Science in the Age of Big Data)*; Ott, M., Pietsch, W., Wernecke, J., Eds.; Springer: New York, NY, USA, 2017; pp. 453–475.
49. Zenil, H.; Kiani, N.A.; Tegnér, J. A Review of Graph and Network Complexity from an Algorithmic Information Perspective. *Entropy* **2018**, *20*, 551. [CrossRef]